SHORT TERM LOAD FORECASTING USING A NEURAL NETWORK BASED TIME SERIES APPROACH

By

SUCI DWIJAYANTI

Bachelor of Science in Electrical Engineering
Sriwijaya University
Palembang, Indonesia
2006

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2013

SHORT TERM LOAD FORECASTING USING A NEURAL NETWORK BASED TIME SERIES APPROACH

Thesis Approved:

Dr. Martin Hagan

Thesis Advisor

Dr. Carl Latino

Dr. R. G. Ramakumar

ACKNOWLEDGMENTS

It gives me great pleasure in expressing my gratitude to all those people who have supported me and had their contributions in making this thesis possible. First and foremost, I must acknowledge and thank The Almighty Allah SWT for blessing, protecting and guiding me throughout this period. I could never have accomplished this without the faith I have in the Almighty.

I am indebted and would like to express my deepest gratitude to my advisor, Dr. Martin Hagan, for his excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research. I would also like to thank Dr. Carl Latino and Dr. R.G. Ramakumar for serving on my committee, for their support and advisement. While doing this thesis, I would never have been able to finish it without help from friends and support from my family. I would like to thank my friend Leni for her help in obtaining the data from PT. PLN Batam Indonesia and all of my friends for their help during the time I am here. Last and most importantly, I would also like to thank my parents, Rahman HS and Suwasti, my sister, Agustin Darmayanti, and brothers, Ridho Akbar and Norman Wibowo, for their endless love, support and encouragement.

Name: SUCI DWIJAYANTI

Date of Degree: May, 2013

Title of Study: SHORT TERM LOAD FORECASTING USING A NEURAL NETWORK BASED TIME SERIES APPROACH

Major Field: Electrical Engineering

Short term load forecasting (STLF) is important, since it is used to maintain optimal performance in the day-to-day operation of electric utility systems. The autoregressive integrated moving average (ARIMA) model is a linear prediction method that has been used for STLF. However, it has a weakness. It assumes a linear relationship between current and future values of load and a linear relationship between weather variables and load consumption. Neural networks have the ability to model complex and nonlinear relationships. Therefore, they can be used as a robust method for nonlinear prediction, and they can be trained with historical hourly load data. The purpose of this work is to describe how neural networks can transform linear ARIMA models to create short term load prediction tools. This thesis introduces a new neural network architecture - the periodic nonlinear ARIMA (PNARIMA) model. In this work, first, we make linear predictions of the daily load using ARIMA models. Then we test the PNARIMA predictor. The predictors are tested using load data (from May 2009 - April 2011) from Batam, Indonesia. The results show that the PNARIMA predictor is better than the ARIMA predictor for all testing periods. This demonstrates that there are nonlinear characteristics of the load that cannot be captured by ARIMA models. In addition, we demonstrate that a single model can provide accurate predictions throughout the year, demonstrating that load characteristics do not change substantially between the wet and dry seasons of the tropical climate of Batam, Indonesia.

TABLE OF CONTENTS

Chapter Page

1 INTRODUCTION 1

2 SHORT TERM LOAD FORECASTING 5

2.1 Overview of Load Forecasting 5

2.2 The importance for short term load forecasting 8

2.3 Short term load forecasting methods 11

2.3.1 Conventional or classical approaches 11

2.3.2 Computational intelligence based techniques 15

3 ARIMA MODELING 20

3.1 Linear time series Overview 20

3.1.1 Stationary stochastic processes 20

3.1.2 Autocovariance and autocorrelation function 21

3.1.3 Differencing 22

3.1.4 White noise 23

3.2 AR, MA, ARMA and ARIMA Models 23

3.3 Time series models for prediction 27

3.3.1 Model identification 27

3.3.2 Parameter estimation 33

3.3.3 Diagnostic testing 34

3.3.4 Forecast 35

4 NEURAL NETWORKS FOR FORECASTING **38**

4.1 Neural Networks Overview 38

4.2 Designing neural networks for forecasting 44

4.2.1 Data Collection and pre-processing 45

4.2.2 Selecting the network type and architecture 46

4.2.3 Selecting training algorithm 47

4.2.4 Analyzing network performance 52

5 RESULTS **54**

5.1 Data Description 54

5.2 Fitting the ARIMA model 59

5.2.1 Preliminary model structure determination 60

5.2.2 Parameter estimation and model validation 64

5.3 Fitting the neural network model 68

5.3.1 Preliminary structure determination 68

5.3.2 Form of neural network prediction 71

5.3.3 Fitting and validation of the neural network model 75

5.4 Comparison of ARIMA and neural network models on test data 77

6 CONCLUSIONS **80**

REFERENCES **83**

APPENDIX **89**

LIST OF TABLES

Table Page

3.1 Behaviour of Theoretical ACF and PACF for Stationary Process 30

3.2 GPAC Array 32

3.3 GPAC Array for An ARMA(p, q) Process 33

5.1 Classification and Composition of Consumers 55

5.2 Training and Testing Periods 58

5.3 Confidence Intervals for Final ARIMA Model Training Data August to October 2009 64

5.4 ARIMA Models 67

5.5 Seasonal ARIMA Performance for Training data 68

5.6 PNARIMA Performance for Training data 77

5.7 ARIMA and PNARIMA Performance for Testing Data 79

LIST OF FIGURES

Figure Page

2.1 A Typical Configuration of An Electric Power System 6

2.2 An Input-Output Configuration of A STLF System and Its Major Uses . . . 10

3.1 System Identification Steps 27

3.2 ACF and PACF of $z_t = 0.5z_{t-1} + e_t$ 29

3.3 ACF and PACF of $z_t = e_t + 0.8e_{t-1}$ 30

4.1 General Neuron 39

4.2 Layer of S Neurons 39

4.3 Multi Layer 40

4.4 Tapped Delay Line 40

4.5 NARX Neural Network 41

4.6 NARX Neural Network Architecture 42

4.7 ARMA Model 42

4.8 Abbreviated Notation of the ARMA Model 43

4.9 Nonlinear ARMA (NARMA) Predictor Model Using a Neural Network . . 43

4.10 Periodic NARIMA (PNARIMA) Model 44

4.11 Flowchart of Neural Network Training Process 44

4.12 Neural Networks 48

4.13 Levenberg-Marquardt Flowchart 49

5.1 Hourly Electric Load Consumption for August 3 - 9, 2009 56

5.2 Peak Load for Each Months from May 2009 until April 2011 56

5.3 Relationship Between Temperature and Daily Peak Electric Load Consumption for August, 2009 57
5.4 ACF and PACF from Original Load August to October, 2009 60
5.5 ACF Differenced of Load August to October, 2009 61
5.6 ACF and PACF Load August - October 2009 Difference $\nabla_1\nabla_{24}$ 62
5.7 ACF and PACF Residual for Tentative Model 63
5.8 ACF and PACF Residual for Final Model 65
5.9 ARIMA Prediction for One Week in August - October, 2009 66
5.10 PNARIMA Architecture for Prediction 70
5.11 ACF and PACF Residual for PNARIMA Neural Network 75
5.12 PNARIMA Prediction During a Typical Week During August - October, 2009 76
5.13 PNARIMA and ARIMA Predictions of Testing Data from The Period from August to October, 2010 78

LIST OF ABBREVIATIONS AND SYMBOLS

List of Abbreviations

ACF	Autocorrelation function
PACF	Partial autocorrelation function
GPAC	Generalized partial autocorrelation
STLF	Short term load forecasting
MTLF	Mid term load forecasting
LTLF	Long term load forecasting
ARMA	Autoregressive moving average
ARIMA	Autoregressive integrated moving average
ARMAX	Autoregressive moving average with exogenous input
NARIMA	Nonlinear autoregressive integrated moving average
PNARIMA	Periodic nonlinear autoregressive integrated moving average
NARX	Nonlinear autoregressive with exogenous input
TDL	Tapped delay line
RMSE	Root mean square error
MAE	Mean absolute error

List of Symbols

z_t	Power load of time t
$\hat{z}_t$	Predictions of power load z_t
$B^n z_t$	Backward shift operator z_{t-n}
∇^d	Difference operator $(1-B)^d$
ϕ_p, θ_q	The autoregressive and moving average parameters
e_t	Error (difference between actual and predicted value)
w_t	The differenced load at time t
$\hat{w}_t$	Predictions of the differenced load w_t
$\mathbf{p}^l(t)$	The i^{th} input vector to the network at time t
$\mathbf{n}^m(t)$	The net input for layer m
$\mathbf{f}^m(.)$	The transfer function for layer m
$\mathbf{a}^m(t)$	The output for layer m
$\mathbf{IW}^{m,l}$	The input weight between input l and layer m
$\mathbf{LW}^{m,l}$	The layer weight between layer l and layer m
$\mathbf{b}^m$	The bias vector for layer m
$DL_{m,l}$	The set of delays in the tapped delay line between Layer l and Layer m
$DI_{m,l}$	The set of delays in the tapped delay line between Input l and Layer m
S^k	Number of neurons in layer k
R	Dimension of input vector
$\|\|w\|\|$	Norm
$\mathbf{W}^k(t)$	Weight matrix
${}_i\mathbf{w}^k(t)$	Weight row vector

CHAPTER 1

INTRODUCTION

Load forecasting plays an important role as a central and integral process in the planning and operation of electric utilities. If the load forecasting is accurate, there will be a great potential savings in the control operations and decision making, such as dispatch, unit commitment, fuel allocation, power system security assessment, and off-line analysis. Errors in forecasting the electric load demand will increase operating costs. Bunn and Farmer [1] pointed out that in the UK, a 1% increase in forecasting error implied a £10 million increase in operating costs. If the predicted electric load is higher than the actual demand, the operating cost will increase significantly, and it wastes scarce resources. On the other hand, if the predicted electric load is less than the actual demand, it can cause brownouts and blackouts, which can be costly, especially to large industrial customers. In addition, reliable load forecasting can reduce energy consumption and decrease environmental pollution.

In general, based on the time horizon, electric load forecasting can be organized into three categories: short term, mid term and long term. In this work, we will focus on short term load forecasting (STLF). STLF refers to the prediction of loads for time leads from one hour up to one week ahead. Mandal et.al [2] explained that STLF is an important tool in day to day operation and planning activities of the utility system, such as energy transactions, unit commitment, security analysis, economic dispatch, fuel scheduling and unit maintenance.

STLF is a very complex process, because there are many factors that influence it, such as economic conditions, time, day, season, weather and random effects. Electric load demand itself is a function of weather variables, human social activities and industrial activi-

ties. Hipert et.al [3] explain that short term load forecasting becomes complicated because the load at a given hour depends not only on the load of the previous hour but also the load at the same hour on previous days, and the load at the same hour on the day with the same denomination in the previous week. In addition, the predictor needs to model the relation between the load and other variables, such as weather, holiday activities, etc.

During the last few decades, various methods for STLF have been proposed and implemented. These methods can be classified into two main types; traditional or conventional and computational intelligence approaches. Time series models, regression models and the Kalman filter are some of the conventional methods. Expert system models, pattern recognition models and neural network models are some of the computational intelligence based techniques. Hagan and Klein [4] were the first to use the periodic ARIMA model of Box and Jenkins for STLF. This is a univariate time series model, in which the load is modeled as a function of its past observed values, with daily and weekly cycles accounted for. Papalexopoulos and Hesterberg [5] used the regression model for STLF. The disadvantage of the regression model is that complex modeling techniques and heavy computational efforts are required to produce reasonably accurate results [6]. Other time series approaches are multiplicative autoregressive models, dynamic linear and nonlinear models, threshold autoregressive models and methods based on Kalman filtering.

Another predictor category is the causal model. In this method, the load is modeled as a function of some exogenous factors, especially weather and social variables. Examples of causal models include the Box-Jenkins transfer function model, ARMAX models, nonparametric regression, structural models and curve fitting procedures. Hagan and Klein [7] introduced the Box and Jenkins transfer function model to STLF. Later, Hagan and Behr [8] added a static nonlinearity to the temperature input of that model. Linear and nonlinear STLF using bilinear models have been performed by Zhang [9]. He included temperature effects to increase the accuracy. Another approach to nonlinear STLF is computational intelligence. Expert systems are intelligent methods that have been implemented to forecast

the short term load in the Taiwan power system [10].

Another computational intelligence method involves neural networks. Neural networks have given excellent results in STLF [3]. They have become popular because of their ability to learn complex and nonlinear relationships through training on historical data, which is very difficult with traditional techniques. Adya and Collopy [11] come to two main conclusions, based on their evaluation: they showed that neural networks have the potential for prediction, and research in neural networks must be validated by making comparisons between the neural networks and alternative methods. Zhang et. al [12] reviewed the application of neural networks to load forecasting and showed that neural networks could deal with the large amount of historical load data with nonlinear characteristics, but they ignored the linear relationship among the data. Other research on STLF using neural networks can be found in [2], [3], [6], [13], [14] and [15].

Since the time series approach is good for capturing linear factors, and neural networks are able to model nonlinearities, this work tries to combine the two approaches. The main objective of this research is to demonstrate how neural networks can transform linear ARIMA models to create a new forecasting tool, which can improve the accuracy of STLF.

In this work, we will first make linear predictions of the daily load using ARIMA models. Then, we develop a new nonlinear predictor from the ARIMA model, using neural networks. This model is called the periodic nonlinear ARIMA (PNARIMA) network. This is a new approach to STLF. We demonstrate that it has higher accuracy than the conventional time series approach (i.e. the ARIMA model). As a case study, the STLF methods will be tested using data obtained from Batam, Indonesia. Batam is chosen because it is the major industrial area in Indonesia, with most of the large industries located there. The load data will be provided by PT. PLN (State Electricity Company) Batam. The data set contains hourly electricity consumption from May 2009 to April 2011.

Accurate STLF is very important for industrial areas such as Batam. As the government corporation that supplies electricity needs, PT. PLN Batam has to meet public elec-

tricity demands continuously. In addition, accurate STLF can help to determine the most economic commitment of generation sources consistent with realibility requirements, operational constraints and policies, and physical, environmental, and equipment limitations. STLF can also be used to assess the security of the power system at any point and provides the system dispatcher with timely information [16].

Following this introduction, the thesis is organized as follows: Chapter 2 discusses the basics of STLF. ARIMA modeling is discussed in Chapter 3, including the fundamentals of time series analysis and system identification. Neural networks for forecasting are discussed in Chapter 4. Chapter 5 shows the results of STLF using the proposed approaches: ARIMA and neural network models. The results are compared to judge the robustness of these two methods. Chapter 6 is the last chapter, and it summarizes the results and makes suggestions for future work.

CHAPTER 2

SHORT TERM LOAD FORECASTING

2.1 Overview of Load Forecasting

The electric power system is a real-time energy delivery system. It is different from water or gas systems which are storage systems. The electric power system is called a real-time system because the power is generated, transported and supplied at the moment we turn on the electric switch. In electric power systems, there are three stages in supplying the power from the power plant to the customers. Those are generation, transmission and distribution. Fig. 2.1 shows a typical configuration of an electric power system. A typical configuration of a power system will be different in each region; it depends on the geographical area, the interconnection, the penetration of renewable resources and the load requirements. The electric power system starts from generation. In this process, the power plant generates the electrical energy. To produce the electrical energy, the power plant transforms other sources of energy, such as heat, solar, hydraulic, wind and fossil fuel. Then, in the power station, the energy is transformed to high voltage electrical energy. The high voltage energy will be transmitted through transmission lines. The energy will be transported from distant generating stations where the energy is produced to the load centers. Before being distributed to the consumers, the sub station will transform the high voltage electrical energy to a lower voltage. This lower voltage energy will be distributed to the customers using the distribution line. Radial or ring distribution circuits are examples of networks in the distribution process. Again, this energy will be transformed based on the type of customers, such as industrial, residential or commercial.

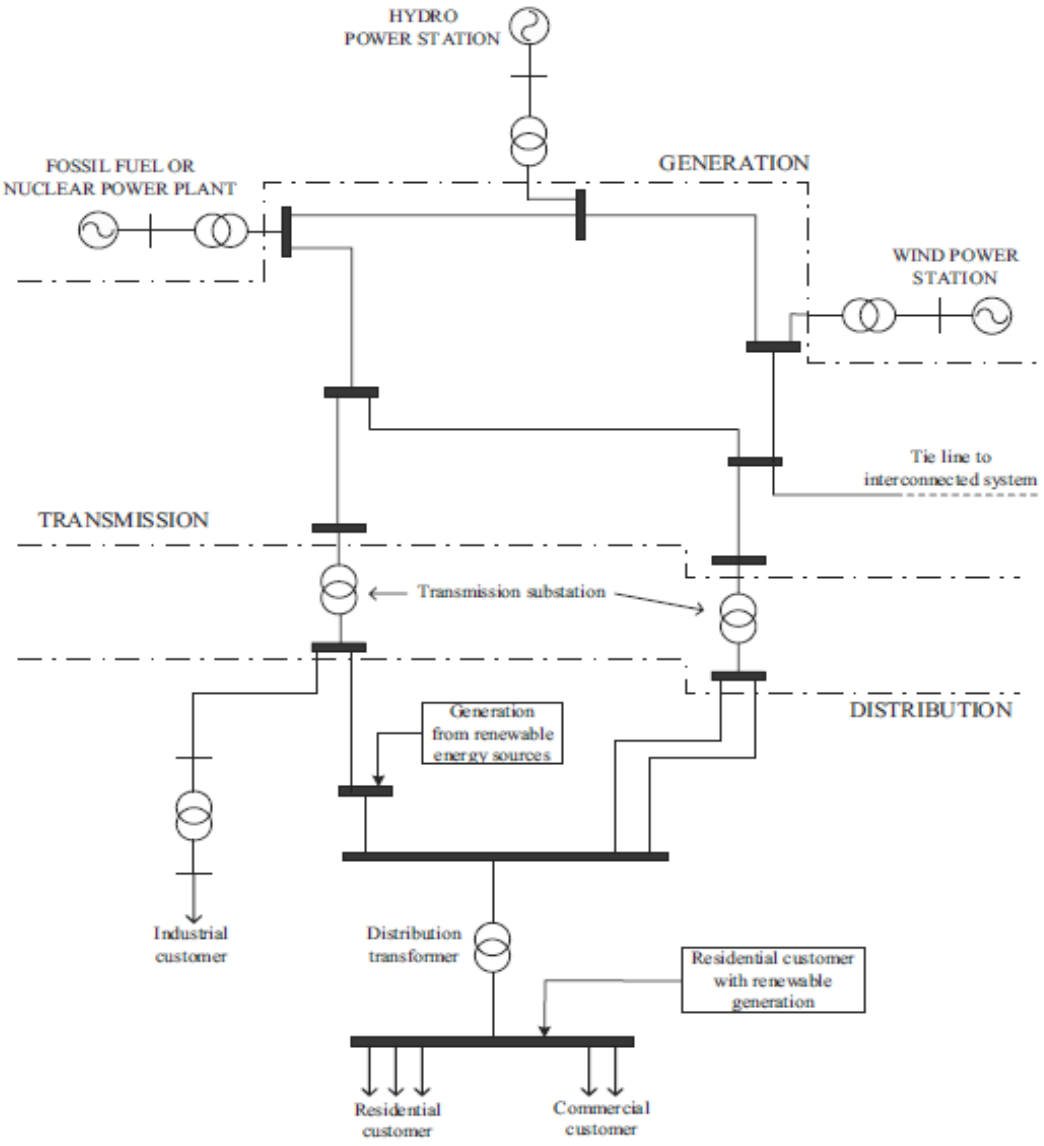


Figure 2.1: A Typical Configuration of An Electric Power System[17]

In the operation of an electric power system, the capability to provide the load to the customers is the most challenging aspect, because it means that they must always fulfill the load requirements instantaneously and at all times. The generator must have extra load that might be used at any time. Due to the significant load fluctuations during each day, the system operator must be able to predict the load demand for the next few hours or even the next few years so that the appropriate planning can be performed. For example, fossil fuel generators need considerable time to be synchronized to the network. This condition forces

the power generation to have available a sufficient amount of generation resources. Hence, prior knowledge of the load requirements enables the electric utility operator to optimally allocate the system resources.

To have prior knowledge of the load requirements, there is a need for load forecasting. The ability to forecast load is one of the most important aspects of effective management of power systems. Load forecasting is essential for planning and operational decision making. Based on the time horizon or lead time, load forecasting can be categorized in three major groups.

1. Short term load forecasting
2. Mid term load forecasting
3. Long term load forecasting

The differences in time horizon have consequences for the models and methods applied and for the input data available and selected. The decision maker must consider not only finding the appropriate model type but also determining the important external factors needed to get the most accurate forecast [18].

Short term load forecasting (STLF) usually forecasts the load up to one week ahead, and is an important tool in such day to day operations of the power system as hydro-thermal coordination, scheduling of energy transactions, estimating load flows and making decisions that can prevent overloading. STLF is an active research area, and there are many different methods. Recently, this area is becoming more and more important because of two main facts: the deregulation of the power systems, which presents new challenges to the forecasting problem, and the fact that no two utilities are the same, which necessitates a detailed case study analysis of the different geographical, meteorological, load type, and social factors that affect the load demand [17]. Typically, there are three mains groups of inputs that are used for STLF. They are seasonal input variables (load variations caused by air conditioning and heating units), weather variables (temperature, humidity, wind and

cloud covers) and historical load data (hourly loads for the previous hour, the previous day, and the same day of the previous week). The output of STLF will be the estimated load every hour in a day, daily or weekly energy generation and the daily peak load.

Another type of load forecasting is mid term load forecasting (MTLF). It has a longer time horizon, from one week to one year. MLTF is used for scheduling maintenance, scheduling of the fuel supply and minor infrastructure adjustments. MLTF also enables a company to estimate the load demand over a longer period, which can help them in negotiations with other companies. Demographic and economic factors influence MLTF. Typically, the output of MLTF is the daily peak and average load [19, 20]. MLTF has a strong relationship with STLF. Longer term decision levels must be incorporated into short term decision levels. This coordination between different decision levels is particularly important in order to guarantee that certain objectives of the operation that arise in the medium-term are explicitly taken into account in the short-term [21]. Moreover, the coordination between decision levels has become an important issue for generation companies in order to increase their profitability.

The last type of load forecasting is long term load forecasting (LTLF). LTLF covers a period of twenty years. LTLF is needed for planning purposes, such as constructing new power stations, increasing the transmission system capacity, and in general for expansion planning of the electric utility. There are more indicators that influence LTLF in demographic and economic development. Some factors that are taken into account in LTLF are population growth, industrial expansion, local area development, the gross domestic product, and past annual energy consumption. The output from this forecasting is the annual peak load demand and the annual energy demand for the years ahead [22]

2.2 The importance for short term load forecasting

STLF is an essential part of daily operations of the utilities. No utility is able to work without it. Moreover, nowadays, STLF has become an urgent matter due to the complexity of

loads, the system requirements, the stricter power quality requirements, and deregulation. The error in forecasting would lead to increased operational cost and decreased revenue. In the deregulation issue, STLF is going to be of benefit in determining the schedule of energy transactions, preparing operational plans and bidding strategies. STLF provides the input data for load flow studies and contingency analysis in case of loss of generator or of line. STLF would be useful for utility engineers in preparing the corrective plan for the different types of expected faults.

STLF is involved in a number of key elements that ensure reliability, security and economic operation of power systems. Gross and Galiana [16] stated the principal objective of the STLF is to provide the load prediction for

1. the basic generation scheduling function to determine the most economic commitment of generation sources consistent with reliability requirements, operational constraints and policies, and physical, environmental, and equipment limitations

2. assessing the security of the power system at any time point, especially to know in which condition the power system may be vulnerable so the dispatchers can prepare the necessary corrective actions such as switching operations, power purchases to operate the systems securely

3. timely dispatcher information to operate the system economically and reliably

To achieve those objectives, some major components are needed. The major components of an STLF system are the STLF model, the data sources, and the man-machine interface. Fig. 2.2 shows a general input-output configuration of an STLF system and its major uses.

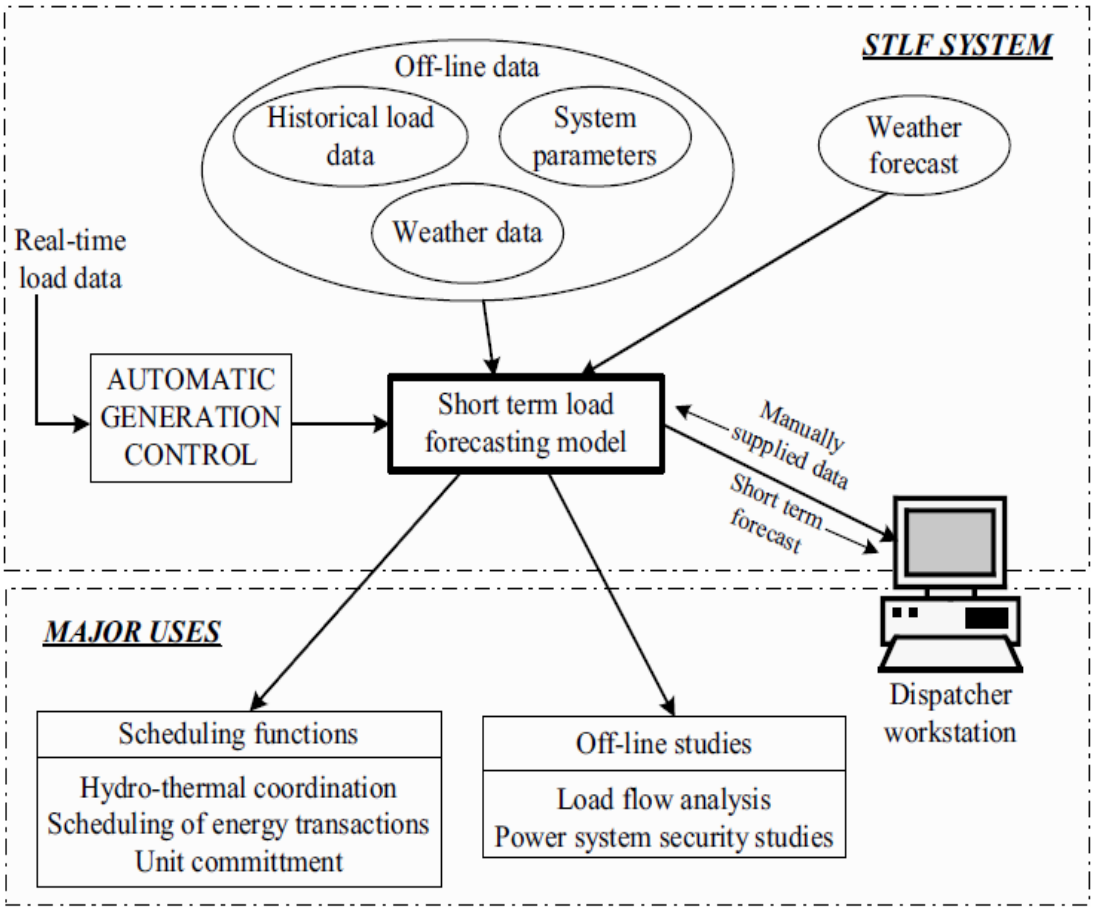


Figure 2.2: An Input-Output Configuration of A STLF System and Its Major Uses [16]

The roles of STLF itself can be divided into three main areas: actions, studies and operations [17]. The role of STLF in actions is that STLF will be an essential part in the negotiation of the bilateral contracts between utilities and regional transmission operator. STLF is needed in studies such as economics dispatch, unit commitment, hydro-thermal coordination, load flow analysis and security studies. In the area of operations, STLF will be used in committing or decommiting generating units and increasing or decreasing the power generation.

2.3 Short term load forecasting methods

There are various approaches to short term load forecasting, which can be classified into two main categories: conventional or classical approaches and computational intelligence approaches.

2.3.1 Conventional or classical approaches

Conventional methods are based on statistical approaches. These require an explicit mathematical model that gives the relationship between the load and another input factors. One of the classical approaches is time series. Time series (univariate) will model the load data as a function of its past observed values. Another approach is a causal model that will represent the load as a function of some exogenous factors, especially weather and social variables. Kyriakides and Polycarpou [17] classify these conventional methods into three categories: time series models, regression models, and Kalman filtering based techniques.

Time series models

Time series models represent the load demand as a function of the previous historical load and assume that the data follow certain stationary patterns, which depend on trends and seasonal variations [23]. In the time series approach, a model is first developed based on previous data, and then future load is predicted based on the model.

There are several time series models used in STLF: ARMA (autoregressive moving average), ARMAX (autoregressive moving average with exogenous variable), ARIMA (autoregressive integrated moving average), ARIMAX (autoregressive integrated moving average with exogenous variables), Box Jenkins, and state space models. ARMA is used for stationary processes, ARIMA is an extension of ARMA for non-stationary processes. In ARIMA and ARMA, load is the only input variable. But since load may also depend on the weather and time of the day, ARIMAX is the most natural tool for load forecasting among

the classical time series models.

One form of time series model that has been suggested [16] is as follows

$$z(t) = y_p(t) + y(t) \tag{2.1}$$

where $y_p(t)$ is a component that depends on the time of the day and on the normal weather pattern for the particular day. The term $y(t)$ is an additive load residual term which describes influences due to weather pattern deviations from normal and random correlation effects. The additive nature of the residual load is justified by the fact that such effects are usually small compared to the time-of-day component. The residual term $y(t)$ can be modeled by an ARMAX process of the form

$$y(t) = \sum_{i=1}^{n} a_i y(t-i) + \sum_{k=1}^{n_u} \sum_{j_k=0}^{m_k} b_{j_k} u_k(t-j_k) + \sum_{h=1}^{H} c_h w(t-h) \tag{2.2}$$

where $u_k(t), k = 1, 2, ..., n_u$ represents the n_u weather-dependent inputs. These inputs are functions of deviations from the normal levels for a given hour of the day of quantities such as temperature, humidity, light intensity, and precipitation. The input $u_k(t)$ may also represent deviations of weather effects measured in different areas of the system. The process $w(t)$ is a zero mean white random process representing the uncertain effects and random load behaviour. The parameters a_i, b_{j_k} and c_h as well as the model order parameters n, n_u, m_k and H are assumed to be constant but unknown parameters to be identified by fitting the simulated model data to observed load and weather data.

Time series models have been implemented to forecast the short term power load. Hagan and Klein [4], [7] used the seasonal ARIMA model and the Box and Jenkins transfer function model for STLF. Hagan and Behr [8] used the Box Jenkins transfer function model with nonlinear temperature transformation, Fan and McDonald described the implementation of ARMA in STLF [24]. Espinoza et.al in their research used the partial autoregressive model. They said that the general problems of STLF and profile identification can be addressed within a unified framework by using the proposed methodology based on the use

of PAR (Partial Autoregressive) models. Starting from a single PAR model template containing 24 seasonal equations, and using the last 48 load values within each equation, it is possible to estimate a model suitable for STLF [25].

In general, time series methods give satisfactory results if there is no change in variables that affect load demand, such as environmental variables. Time series modeling is particularly useful when little knowledge is available on the underlying data generating process or when there is no satisfactory explanatory model that relates the prediction variable to other explanatory variables [26]. In the time series approach, it is assumed that the load demand is a stationary time series and has normal distribution characteristics. The result of forecasting will be inaccurate when there is a change in the variable and when the historical data is not stationary. Although time series models have provided good results in some cases, the approach is limited because of the assumption of linearity.

Regression models

The regression model represents the linear relationship between the load and other influenced variables such as weather, customer types and day type. It uses the technique of weighted least squares estimation using historical data. In this approach, temperature is the most important information for electric load forecasting among weather variables and is usually modeled in a nonlinear form.

Mbamalu and El-Hawary [27] describe a method to forecast short-term load requirements using an iteratively reweighted least squares algorithm. They used the following load model

$$Y_t = v_t a_t + \epsilon_t \tag{2.3}$$

Where t is sampling time, Y_t is measured system total load, v_t is vector of adapted variables, such as time, temperature, light intensity, wind speed, humidity, day type (workday, weekend), etc., a_t is transposed vector of regression coefficients, and ϵ_t is model error at time t. Additional research was performed by Haida and Muto [28]. They used the re-

gression method based on the daily peak load and then combined it with a transformation technique to generate a model that utilizes both the annual weather-load relationship and the latest weather load characteristic.

Regression methods are relatively easy to implement. Its advantage is that the relationship between input and output variables is easy to comprehend and easy in performance assessments. Although regression based methods are widely used by electric utilities, there are some deficiencies of this method due to the nonlinear and complex relationship between the load demand and the influencing factors. Besides that, heavy computational efforts are required to get reasonably accurate results. The main reason for the drawbacks is that the model is linearized in order to get the estimated coefficients. However, the load patterns are nonlinear and it is impossible to represent the load demand during distinct time periods using a linearized model.

Kalman filtering based techniques

The Kalman filter is an algorithm for adaptively estimating the state of the model. In load forecasting, the input-output behavior of the system is represented by a state-space model with the Kalman filter used to estimate the unknown state of the model. So, the Kalman filter uses the current prediction error and the current weather data acquisition programs to estimate the next state vector.

There has been research that uses Kalman filtering for STLF. Park et. al [29] developed a state model for the nominal load which consists of three components: nominal load, type load and residual load. The nominal load is modeled such that the Kalman filter can be used, and the parameters of the model are adapted by the exponentially weighted recursive least-squares method. The effect of weekend days is represented through the type load model, which is added to the nominal load estimated through Kalman filtering. Type load is determined through exponential smoothing. To account for modeling error, residual load is also calculated. Another technique for STLF using the Kalman filter was implemented

by Al-Hamadi and Soliman [30]. They used Kalman filter-based estimation to estimate the model parameters using historical load and weather data. Sagunaraj [31] used the Kalman filtering algorithm with the incorporation of ”a fading memory”. A two stage forecast is carried out, where the mean is first predicted, and a correction is then incorporated in real time using an error feedback from previous hours. They implemented this method for developing countries where the total load is not large.

Although the Kalman filter has been used in STLF, it has some limitations. One of the key difficulties is to identify the state space model parameters.

2.3.2 Computational intelligence based techniques

To improve the STLF performance that can be obtained using conventional approaches, researchers have recently turned their focus to computational intelligence (CI), which is getting more and more popular. In this method, no complex mathematical formulation or quantitative correlation between inputs and outputs are required. CI techniques have the potential to give better forecasting accuracy. Accuracy is the key in load forecasting. For every small decrease in forecasting error, the operating savings are considerable. It is estimated that a 1% decrease in forecasting error for a 10 GW electric utility can save up to 1.6% million annually [32].

Artificial neural network

Artificial neural networks (ANN) have been implemented in many applications because of their ability to learn. ANNs are based on biological neurons, and are frequently applied for load forecasting. The idea for using neural networks for forecasting is the assumption that there exists a nonlinear function that relates some external variable to future values of the time series. Kyriakides and Polycarpou [17] said that there are three steps that need to be considered in using neural network models for time series predictions: (i) designing the neural network model; e.g, choosing the type of neural network that will be employed,

the numbers of layers and the number of adjustable parameters or weights, (ii) training the neural network; this includes selecting the training algorithm, the training data that will be used and also the pre-processing of the data, (iii) testing the trained network on a data set that has not been used during the training stage (typically referred to as neural network validation).

A feed-forward network, which consists of several successive layers of neurons with one input layer, several hidden layers and one output layer, is most often applied in forecasting. The basic learning or weight-adjusting procedure is back propagation (a form of steepest descent), which propagates the error backwards and adjusts the weight accordingly. There is much research that has used neural networks for STLF. Peng et.al [13] used an adaptive linear combiner, called an "ADALINE" to forecast the load one week ahead. Senjyu et. al [14] proposed a neural network for one hour ahead load forecasting by using the correction of similar day data. In the proposed prediction method, the forecasted load is obtained by adding a correction to the selected similar day data. Adya and Collopy [11] investigated 48 studies using neural networks to see their effectiveness for forecasting. Hippert et.al [3] also examined the application of neural networks, but they made it more specific to short term load forecasting.

From the studies that have been done, it can be concluded that neural networks have potential as a load forecasting tool, due to their nonlinear approximation capabilities and the availability of convenient methods for training. By learning from training data, neural networks extract the nonlinear relationship among the input variables. Neural networks have ability to model multivariate problems without making complex dependency assumptions among input variables. But neural networks also have some limitations. They are not typically able to handle significant uncertainty or to use "common sense knowledge" and perform accurate forecasts in abnormal situations. Sometimes, the researchers combine this method with conventional techniques to overcome some drawbacks of the original method. Zhang [26] made a hybrid between an ARIMA and a neural network model. Zhao

and Su [33] used a Kalman filter and an Elman neural network.

Expert systems

An expert system is a computational model that comprises four main parts: a knowledge base, a data base, an inference mechanism and a user interface. The knowledge base is a set of rules that are derived from the experience of human experts. The data base is a collection of facts obtained from the human experts and information obtained through the interference mechanism of the system. The inference mechanism is the part of the expert system that ”thinks”. In load forecasting, the expert system uses some rules which are usually heuristic in nature to get accurate forecasting. The expert system will transform the rules and procedures used by the human experts to the software that has the capability to forecast automatically without human assistance. This brings advantages since expert systems can make decisions when the human experts are unavailable. They can reduce the work burden of human experts and make fast decisions in case of emergencies.

To get the best result by using expert systems, there must be a collaboration between the availability of the human expert and the software developers. This is because the time imparting the expert's knowledge to the expert system software must be considered in making the system software. STLF using expert systems was proposed by Ho et.al [10]. Here, the case study was the Taiwan power system. The operators knowledge and the hourly observations of system load over the past five years were employed to establish eleven day types. Weather parameters were also used. The other research was done by Rahman and Hazim [34]. They developed a site-independent technique for STLF. Knowledge about the load and the factors affecting it are extracted and represented in a parameterized rule base.

Fuzzy logic

A generalization of the usual Boolean logic used for digital circuit design is known as fuzzy logic. Under fuzzy logic, an input is associated with certain qualitative ranges. The benefit

of fuzzy logic is that there is no need to make a mathematical model mapping inputs to outputs and no need to have precise inputs. Hence, properly designed fuzzy logic systems can be used in forecasting and will be robust. After the logical processing of fuzzy inputs, a ”defuzzification” process can be used to gain the precise outputs.

Kiartzis and Bakirtzis [35] used a fuzzy expert system to forecast daily load curves with two minima and two maxima, for each season of the year. The inference operations of the fuzzy rules are performed following the Larsen Max-Product implication method and the Product Degree of Fulfillment method, while the defuzzification procedure is based on the Center of Area method. The proposed fuzzy expert system for peak load forecasting is tested using historical load and temperature data of the Greek interconnected power system. Sometimes fuzzy is combined with the neural network as a hybrid method. This hybrid method has some advantages, such as the ability to respond accurately to unexpected changes in the input variables, the ability to learn from experience and the ability to synthesize new relationships between the load demand and the input variables [17]. Srinivasan et.al [36] developed and implemented a hybrid fuzzy neural based one-day ahead load forecaster. The approach involves three main stages. In the first stage, historical load was updated to the current load demand by studying the growth trend and making the necessary compensation. The second stage attempts to map the load profile of the different days by means of Kohonen’s self organizing map. The load forecast for the current day is then obtained using the auto-associative memory of the neural network. A fuzzy parallel processor takes variables such as day type, weather and holiday proximity into consideration when making the required hourly load accommodations for each day.

Support vector machine

The support vector machine performs a nonlinear mapping of the data by using kernel functions to transform the original data into a high dimensional space and then does linear regression in this high dimensional space. In the other words, linear regression in a high

dimensional feature space corresponds to a nonlinear regression in the low dimensional input space. Once the transformation is achieved, optimization techniques are used to solve a quadratic programming problem, which yields the optimal approximation parameters.

Typically, support vector machines are usually used for data classification and regression. Instead of performing the regression in the original (x, y) - space the x data are mapped into a higher dimensional space using a mapping function. In the context of load forecasting, the radial basis function (RBF) kernel is used in most cases. The RBF can be expressed as

$$\phi(x_i)^T \phi(x_j) = \exp(-\gamma |x_i - x_k|^2) \tag{2.4}$$

Espinoza et.al [25] applied this technique by using fixed size Least Squares Support Vector Machines (LS-SVM). The methodology is applied to the case of load forecasting as an example of a real-life large scale problem in industry, for the case of 24-hour ahead predictions based on the data from a sub-station in Belgium. Other studies were done by Chen et al. [37]. They proposed the SVM model to predict the daily load demand in a month. They won the EUNITE competition. In their study, they identified two clearly separate patterns for summer and winter load time series.

CHAPTER 3

ARIMA MODELING

3.1 Linear time series Overview

3.1.1 Stationary stochastic processes

A time series is a set of observations ordered in time or any other dimension. There are two types of time series; deterministic and stochastic. A process is deterministic if future behaviour can be exactly predicted. Whereas for a stochastic process past knowledge can only indicate the probabilistic structure of future behaviour. To be more precise, a stochastic process $Z(t)$, for $t \in T$, can be defined as a collection of random variables, where T is an index set. When T represents time, the stochastic process is referred to as a time series [38].

A special case of stochastic process is the stationary process, which is in a particular state of statistical equilibrium. A time series is strictly stationary if its properties are not affected by a change in the time origin. In other words, a stochastic process is strictly stationary if the joint distribution of the observations $z_{t_1}, z_{t_2}, ..., z_{t_m}$ is exactly the same as the joint distribution of the observations $z_{t_1+k}, z_{t_2+k}, ..., z_{t_m+k}$. The stationary assumption implies that the joint probability distribution $p(z_t)$ is the same for all times t and may be written as $p(z)$ when $m = 1$. Thus, the stochastic process has a constant mean

$$\mu = E[z_t] = \int_{-\infty}^{\infty} z p(z)\, \mathrm{d}z \tag{3.1}$$

and also a constant variance

$$\sigma_z^2 = E[(z_t - \mu)^2] = \int_{-\infty}^{\infty} (z - \mu)^2 p(z)\, \mathrm{d}z \tag{3.2}$$

To estimate these parameters, we can use the sample mean and the sample variance. If the observations in the time series are $z_1, z_1, ..., z_N$ then the sample mean can be estimated as

$$\hat{\mu} = \frac{1}{N}\sum_{t=1}^{N} z_t \tag{3.3}$$

and the sample variance can be estimated as

$$\hat{\sigma}_z^2 = \frac{1}{N-1}\sum_{t=1}^{N}(z_t - \hat{\mu})^2 \tag{3.4}$$

3.1.2 Autocovariance and autocorrelation function

As described above, if a time series is stationary, the joint probability distribution $p(z_{t_1}, z_{t_2})$ is the same for all times t_1 and t_2 that are separated by the same interval. The autocovariance function of a stationary process can be defined by

$$R(k) = E[(z_t)(z_{t+k})] \quad \text{for } k = 0, \pm 1, \pm 2, \ldots \tag{3.5}$$

If the mean is constant, and the autocorrelation is only a function of lag k, then the series is wide sense stationary.

Besides autocovariance, the autocorrelation function can also be defined for stationary processes. The autocorrelation function (ACF) is defined as

$$\rho(k) = \frac{R(k)}{R(0)} \tag{3.6}$$

The estimation of $R(k)$ is obtained by

$$\hat{R}(k) = \frac{1}{N}\sum_{t=1}^{N-|k|}(z_t)(z_{t+|k|}) \quad \text{for } k = 0, \pm 1, \pm 2, \ldots, \pm(N-1) \tag{3.7}$$

Another tool that will be needed in time series modeling is partial autocorrelation function (PACF). The PACF is defined as the correlation between two variables after being adjusted for a common factor that may be affecting them. The PACF between z_t and z_{t-k} is the autocorrelation between z_t and z_{t-k} after adjusting for $z_{t-1}, z_{t-2}, \ldots, z_{t-k+1}$. The

PACF is denoted by $\{\phi_{kk} : k = 1, 2, \ldots\}$. Let us consider a stationary time series $\{z_t\}$. For any fixed value of k, the Yule-Walker equations for the PACF of an AR(p) process is

$$\rho(j) = \sum_{i=1}^{k} \phi_{ik}\rho(j-1),\ j = 1, 2, \ldots, k$$

It can be written in matrix form

$$\begin{bmatrix} 1 & \rho(1) & \rho(2) & \ldots & \rho(k-1) \\ \rho(1) & 1 & \rho(3) & \ldots & \rho(k-2) \\ \rho(2) & \rho(1) & 1 & \ldots & \rho(k-3) \\ \vdots & & & & \\ \rho(k-1) & \rho(k-2) & \rho(k-3) & \ldots & 1 \end{bmatrix} \begin{bmatrix} \phi_{1k} \\ \phi_{2k} \\ \vdots \\ \phi_{kk} \\ \end{bmatrix} = \begin{bmatrix} \rho(1) \\ \rho(2) \\ \rho(3) \\ \vdots \\ \rho(k) \end{bmatrix} \tag{3.8}$$

or

$$\mathbf{P_k}\phi_{\mathbf{k}} = \rho_{\mathbf{k}}$$

Thus to solve for ϕ_k, we have

$$\phi_k = \mathbf{P_k^{-1}}\rho_{\mathbf{k}} \tag{3.9}$$

For any given k, $k = 1, 2, \ldots$, the last coefficient ϕ_{kk} is the PACF. The sample PACF, $\hat{\phi}_{kk}$, is obtained by using the sample ACF, $\hat{\rho}(k)$.

3.1.3 Differencing

Most time series are characterized by a trend. Trends indicate a non-stationary time series. There are several approaches to remove the trend, such as regression models and differencing. The second approach is suggested by Box and Jenkins [39]. The method of differencing a time series consists of subtracting the values of observations from one another in some prescribed time-dependent order. First, let us define the backward shift

operator as

$$\begin{aligned} Bz_t &= z_{t-1} \\ B^n z_t &= z_{t-n} \end{aligned} \tag{3.10}$$

Using the backward shift operator, the difference operator can be written as

$$\begin{aligned} w_t &= \nabla^d z_t \\ w_t &= (1-B)^d z_t \end{aligned} \tag{3.11}$$

A comparison between polynomial curve fitting and differencing is described in [40]. Differencing has advantages over fitting a trend model to the data. It does not require estimation of any parameters, which makes it a simple approach. The other advantage is that differencing can allow a trend component to change through time. It is not deterministic, like fitting a trend model. Generally, to remove the underlying trend in the data, one or more differences are required.

3.1.4 White noise

If a time series consists of uncorrelated observations and has constant variance, then we call it white noise. White noise is an example of a stationary process. A process $[e_t, t = 0, 1, 2, \ldots]$ is a white noise process if

$$cov(e_t, e_{t+k}) = \begin{cases} \sigma_e^2 & \text{if } k = 0 \\ 0 & \text{otherwise} \end{cases}$$

3.2 AR, MA, ARMA and ARIMA Models

Let $z_t, z_{t-1}, z_{t-2}, ..., z_{t-n}$ be power loads, with $t, t-1, t-2, ..., t-n$ representing integer values of time in hours. Hence the power load z_t can be seen as a time series.

In linear time series analysis, we often represent the process as the output of a linear system driven by white noise:

$$z_t = \sum_{i=0}^{\infty} \psi_i e_{t-i} \tag{3.12}$$

where ψ_i is the system impulse response and e_t is white noise. (We assume here that the process is zero mean.)

Under appropriate conditions [39], the process can also be written as a weighted sum of previous values of the process plus white noise:

$$z_t = \sum_{i=1}^{\infty} \phi_i z_{t-i} + e_t \tag{3.13}$$

In some cases, only a finite number of previous values are needed:

$$z_t = \sum_{i=1}^{p} \phi_i z_{t-i} + e_t \tag{3.14}$$

where p is the process order, which will be determined using system identification techniques. Eq. (3.14) can be expanded as

$$z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + \ldots + \phi_p z_{t-p} + e_t \tag{3.15}$$

Eq. (3.15) is called an autoregressive model (AR). An autoregressive model, AR(p), expresses a time series as a linear function of its past values. The order of the AR model tells how many past values are included. In the AR(p) model, the current value of the process is expressed as a linear combination of p past observations of the process and white noise. For convenience, we can use the backward shift operators from (3.11) to define

$$\phi(B) \quad = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_n B^n \tag{3.16}$$

Hence, (3.15) can be written as

$$\phi(B) z_t = e_t \tag{3.17}$$

In addition to the autoregressive (AR) model, the other important fundamental class of time series is the moving average (MA) model. MA (q) is a model in which the time series

is regarded as a moving average (unevenly weighted) of a white noise series e_t . In the MA(q) model the current value of the process is expressed as a linear combination of q previous values of white noise.

$$\begin{aligned} z_t &= e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \ldots - \theta_q e_{t-q} \\ \theta(B) &= 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_n B^n \\ z_t &= \theta(B) e_t \end{aligned} \tag{3.18}$$

ARMA(p,q) is a mixed model, which is an extension of AR and MA models. The mixed autoregressive-moving average model ARMA(p,q) is a combination of (3.17) and (3.18)

$$\begin{aligned} z_t &= \phi_1 z_{t-1} + \ldots + \phi_p z_{t-p} + e_t - \theta_1 e_{t-1} - \ldots - \theta_q e_{t-q} \\ \phi(B) z_t &= \theta(B) e_t \end{aligned} \tag{3.19}$$

This ARMA model, introduced by Box Jenkins [39], has become one of the most popular models for forecasting. The ARMA model (3.19) can be used to model stationary processes with finite variance, and it is assumed that the roots of $\theta(B)$ and $\phi(B)$ lie outside the unit circle.

Some processes may show non-stationarity because the roots of $\phi(B) = 0$ lie on the unit circle. In particular, non-stationary series are often well represented by models in which one or more of these roots are unity. Now, let us consider

$$\varphi(B) z_t = \theta(B) e_t \tag{3.20}$$

where $\varphi(B)$ is a non-stationary autoregressive operator, where d of the roots of $\varphi(B) = 0$ are unity and the remainder lie outside the unit circle. Eq. (3.20) can be expressed as

$$\varphi(B) z_t = \phi(B)(1 - B)^d z_t = \phi(B) \nabla^d z_t = \theta(B) e_t \tag{3.21}$$

where $\phi(B)$ is a stationary autoregressive operator. Equivalently, the process is defined by

$$\phi(B) w_t = \theta(B) e_t \tag{3.22}$$

where $w_t = \nabla^d z_t$. This process of differencing results in an autoregressive-integrated-moving average model ARIMA(p,d,q). The value of p, d, q can be estimated using the autocorrelation function (ACF) and the partial autocorrelation function (PACF).

Standard ARIMA models cannot really cope with seasonal behaviour. To incorporate seasonal behaviour, we can use the general seasonal ARIMA $(p, d, q) \times (P, D, Q)_s$ model, as follows

$$\phi_p(B)\Phi_P(B^s)\nabla^d\nabla_s^D z_t = \theta_q(B)\theta_Q(B^s)e_t \tag{3.23}$$

where s is the period of the seasonal pattern. The power load requires a seasonal model because of the periodic nature of the load curve (e.g. The load at 10 A.M Tuesday is related to the load at 10 A.M Monday). It is advantageous to use the seasonal ARIMA $(p, d, q) \times (P, D, Q)_{24}$ models:

$$\phi_p(B)\Phi_P(B^{24})\nabla^d\nabla_{24}^D z_t = \theta_q(B)\theta_Q(B^{24})e_t \tag{3.24}$$

In some cases, it is also useful to recognize the weekly periodicity (Sundays are not like Mondays) and a two period ARIMA $(p, d, q) \times (P, D, Q)_{24} \times (P^{`}, D^{`}, Q^{`})_{168}$ could be used

$$\phi_p(B)\Phi_P(B^{24})\Phi_P^{`}(B^{168})\nabla^d\nabla_{24}^D\nabla_{168}^{D^{`}} z_t = \theta_q(B)\theta_Q(B^{24})\theta_Q^{`}(B^{168})e_t \tag{3.25}$$

The ARIMA models are essentially extrapolations of the previous load history and have problems when there is a sudden change in the weather. The transfer function model allows for the inclusion of some independent weather variables, such as temperature. The transfer function model TRFU(r,s) is

$$z_t = \frac{\omega(B)}{\delta(B)}x_{t-b} + n_t \tag{3.26}$$

Where $\omega(B)$ is a polynomial in B of order s, and $\delta(B)$ is a polynomial in B of order r. the disturbance process n_t is not white, but can be represented by an ARIMA model

$$\nabla^d\nabla_{24}^D\nabla_{168}^{D^{`}} z_t = \frac{\omega(B)}{\delta(B)}\nabla^d\nabla_{24}^D\nabla_{168}^{D^{`}} x_{t-b} + \frac{\theta_q(B)\Theta_Q(B^{24})\Theta_{Q^{`}}^{`}(B^{168})}{\phi_p(B)\Phi_P(B^{24})\Phi_{P^{`}}^{`}(B^{168})}e_t \tag{3.27}$$

3.3 Time series models for prediction

The model that will be used for short term load forecasting is the ”seasonal” ARIMA model. Before making prediction, we need to build an ARIMA model based on measured data. This is a three-step iterative procedure

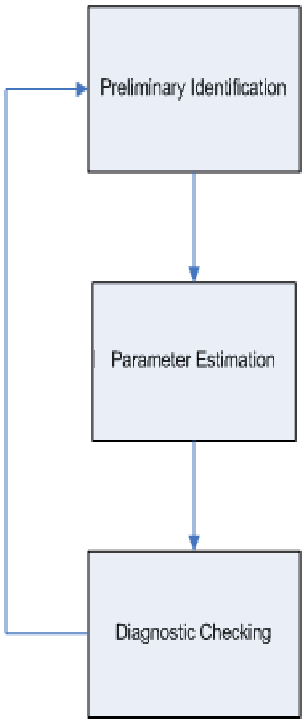


Figure 3.1: System Identification Steps

First, a tentative model of the ARIMA class is identified through analysis of the historical data. Then, the unknown parameters of the model are estimated. The next step is to perform diagnostic checks to determine the adequacy of the model and to indicate potential improvements. If the model is not adequate, then the modeling process will be restarted from the beginning. This will be repeated until the final model is adequate.

3.3.1 Model identification

Modeling begins with preliminary identification. The purpose of the preliminary step is to determine appropriate orders for the ARIMA model. This is done by analyzing the

autocorrelation function (ACF) and partial autocorrelation function (PACF).

From the ACF, we can determine the stationarity of the process. For a stationary time series, the ACF will typically decay rapidly to 0. But in non-stationary time series, the ACF will typically decay slowly, because the observed time series presents trends and heteroscedasticity. If the process is not stationary, data transformations (i.e differencing and power transformations) are needed to make the time series stationary. They will remove the trend and stabilize the variance before an ARIMA model can be fitted. Once stationarity can be presumed, the ACF and PACF of the stationary time series are analyzed to determine the order of the time series model.

For an AR (p) process, the autocorrelation function can be found as

$$\begin{aligned} R(k) &= E[z_t z_t - k] \\ R(k) &= \phi_1 E[z_{t-1} z_{t-k}] + \phi_2 E[z_{t-2} z_{t-k}] + \ldots + \phi_p E[z_{t-p} z_{t-k}] + E[e_t z_{t-k}] \\ R(k) &= \phi_1 R(k-1) + \phi_2 R(k-1) + \ldots + \phi_p R(k-p),\ k > 0 \end{aligned} \tag{3.28}$$

The expectation $E[e_t z_{t-k}]$ vanishes because z_{t-k} is uncorrelated with e_t for $k > 0$. If we divide both sides by R(0), we obtain the normalized ACF

$$\rho(k) = \phi_1 \rho(k-1) + \phi_2 \rho(k-2) + \ldots + \phi_p \rho(k-p) \tag{3.29}$$

To see the behaviour of the ACF of an AR(p) process, let us take an AR(1) example. Eq. (3.29) will become

$$\rho(k) = \phi_1 \rho(k-1) \ \text{ and } \ |\phi_1| < 1, k > 0$$

Hence for AR(1), the autocorrelation becomes

$$\rho(k) = \phi_1^k$$

Therefore, $\rho(k)$ decreases exponentially as the lag k increases. In a general ARMA process, $\rho(k)$ is a combination of damped sinusoids and exponentials.

Let ϕ_{kj} denote the j^{th} coefficient in a fitted AR model of order k. The last coefficient, ϕ_{kk}, is the PACF. For an AR(p) process we can show that

$$\phi_{kk} \begin{cases} \neq 0 & \text{if } k \leq p \\ = 0 & \text{otherwise} \end{cases}$$

Fig. (3.2) shows the characteristics of the sample ACF and PACF for an AR(1) process.

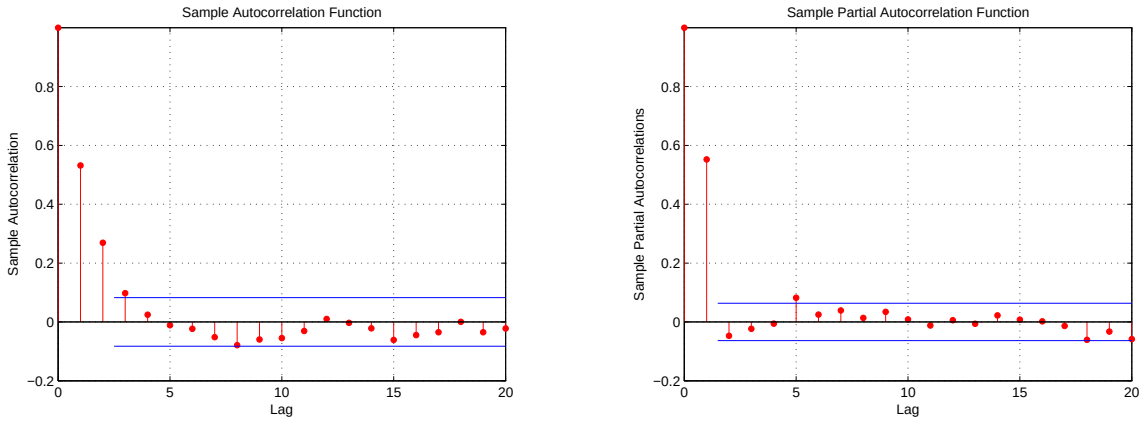


Figure 3.2: ACF and PACF of $z_t = 0.5z_{t-1} + e_t$

For an MA(q) process, the ACF is identically zero for lags k greater than q. An MA(q) process can be equally written as

$$\theta^{-1}(B)z_t = e_t \tag{3.30}$$

where $\theta(B)$ is assumed to be invertible, with its inverse denoted by $\theta^{-1}(q)$. Therefore, the PACF in an MA process has infinite components. Fig. (3.3) illustrates estimated ACF and PACF for an MA(1) process

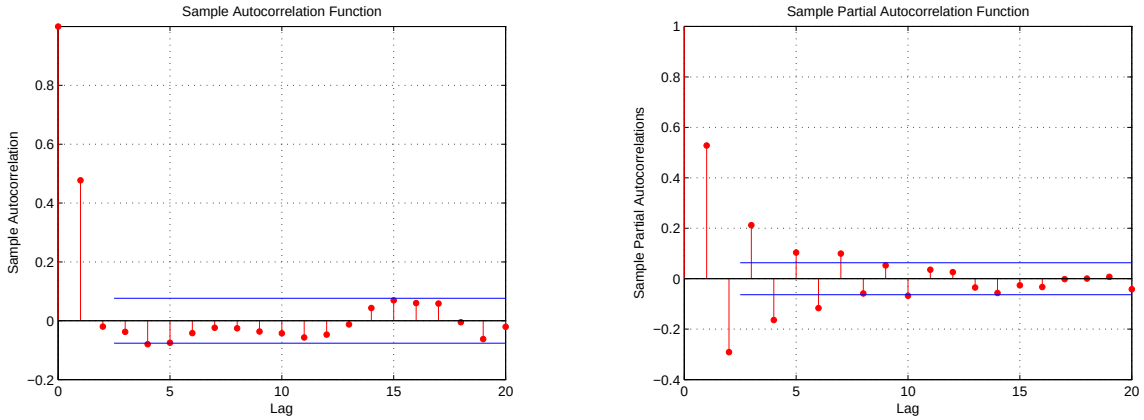


Figure 3.3: ACF and PACF of $z_t = e_t + 0.8e_{t-1}$

Clearly, AR and MA processes follow different patterns of ACF and PACF. Table 3.1 below can be used as a guide to identify AR or MA models based on the pattern of the ACF and PACF.

Table 3.1: Behaviour of Theoretical ACF and PACF for Stationary Process

Model	ACF	PACF
MA(q)	Cuts off after lag q	Exponential decay and/or damped sinusoid
AR(p)	Exponential decay and/or damped sinusoid	Cuts off after lag p
ARMA(p,q)	Exponential decay and/or damped sinusoid	Exponential decay and/or damped sinusoid

One major concern about the ACF and PACF approach in the case of an ARMA(p,q) process with both $p > 0$ and $q > 0$ is that there is uncertainty concerning model selection when examining only the ACF and PACF. Because of this, Woodward and Gray in 1981 defined the generalized partial autocorrelation (GPAC). First, let z_t be a stationary process

with ACF $\rho_j, j = 0, \pm1, \pm2, \ldots$, and consider the following $k \times k$ system of equations

$$\begin{aligned}
\rho_{j+1} &= \phi_{k1}^{(j)}\rho_j + \phi_{k2}^{(j)}\rho_{j-1} + \ldots + \phi_{k,k-1}^{(j)}\rho_{j-k+2} + \phi_{kk}^{(j)}\rho_{j-k+1} \\
\rho_{j+2} &= \phi_{k1}^{(j)}\rho_{j+1} + \phi_{k2}^{(j)}\rho_j + \ldots + \phi_{k,k-1}^{(j)}\rho_{j-k+3} + \phi_{kk}^{(j)}\rho_{j-k+2} \\
&\vdots \\
\rho_{j+k} &= \phi_{k1}^{(j)}\rho_{j+k-1} + \phi_{k2}^{(j)}\rho_{j+k-2} + \ldots + \phi_{k,k-1}^{(j)}\rho_{j+1} + \phi_{kk}^{(j)}\rho_j
\end{aligned} \tag{3.31}$$

where $\phi_{ki}^{(j)}$ denotes the ith coefficient associated with the $k \times k$ systems in which ρ_{j+1} is on the left-hand side of the first equation. The GPAC function is defined to be $\phi_{kk}^{(j)}$. By using Cramer's rule, it can be solved with

$$\phi_{kk}^{(j)} = \frac{\begin{vmatrix} \rho_j & \cdots & \rho_{j-k+2} & \rho_{j+1} \\ \rho_{j+1} & \cdots & \rho_{j-k+3} & \rho_{j+2} \\ \vdots & & & \\ \rho_{j+k-1} & \cdots & \rho_{j+1} & \rho_{j+k} \end{vmatrix}}{\begin{vmatrix} \rho_j & \cdots & \rho_{j-k+2} & \rho_{j-k+1} \\ \rho_{j+1} & \cdots & \rho_{j-k+3} & \rho_{j-k+2} \\ \vdots & & & \\ \rho_{j+k-1} & \cdots & \rho_{j+1} & \rho_j \end{vmatrix}} \tag{3.32}$$

The GPAC can uniquely determine the orders p and q of an ARMA(p,q) process when the true autocorrelations are known. For an ARMA(p,q) process, $\phi_{pp}^{(q)} = \phi_p$. Also, as with the partial autocorrelation function, it can be shown that if $k > p$ then $\phi_{kk}^{(q)} = 0$. Thus, the GPAC provides identification of p and q uniquely for an ARMA(p,q) model in much the same way as the partial autocorrelation does for identification of p for an AR(p).

Another useful property of the GPAC is described by Woodward and Gray [41]. Let z_t be an ARMA process with autoregressive order greater than zero, then

1. z_t is an ARMA(p,q) if and only if $\phi_{kk}^{(q)} = 0, k > p$ and $\phi_{pp}^{(q)} \neq 0$

2. if z_t ARMA(p,q) then $\phi_{pp}^{(q+h)} = \phi_q, h \geq 0$

Woodward et.al [41] also recommended examining the GPAC by calculating $\phi_{kk}^{(j)}, k = 1, 2, \ldots P$ and $j = 1, 2, \ldots Q$ for some P and Q then placing these value in a GPAC array, as in Table 3.2.

Table 3.2: GPAC Array

j/k		Autoregressive order			
		1	2	...	P
Moving average order	0	$\phi_{11}^{(0)}$	$\phi_{22}^{(0)}$	...	$\phi_{PP}^{(0)}$
	1	$\phi_{11}^{(1)}$	$\phi_{22}^{(1)}$	...	$\phi_{PP}^{(1)}$
	2	$\phi_{11}^{(2)}$	$\phi_{22}^{(2)}$	...	$\phi_{PP}^{(2)}$
	$\vdots$	$\vdots$	$\vdots$	...	$\vdots$
	Q	$\phi_{11}^{(Q)}$	$\phi_{22}^{(Q)}$	...	$\phi_{PP}^{(Q)}$

The first row of the GPAC array consists of the partial autocorrelations. We need to find a row in which zeros begin occurring beyond a certain point. This row is the qth row, and the zeros begin in the $p + 1$st column. Also, values in the pth column are constant from the qth row and below. This constant is $\phi_p \neq 0$. Given the true autocorrelation for an ARMA(p,q) process, the patterns in the GPAC array uniquely determine the model orders if P and Q are chosen sufficiently large. Table 3.3 shows the GPAC array for an ARMA(p,q) process.

Table 3.3: GPAC Array for An ARMA(p, q) Process

j/k		Autoregressive order						
		1	2	$\ldots$	p	$p+1$	$\ldots$	
Moving average order	0	$\phi_{11}^{(0)}$	$\phi_{22}^{(0)}$	$\ldots$	$\phi_{pp}^{(0)}$	$\phi_{p+1,p+1}^{(0)}$	$\cdots$	
	1	$\phi_{11}^{(1)}$	$\phi_{22}^{(1)}$	$\ldots$	$\phi_{pp}^{(1)}$	$\phi_{p+1,p+1}^{(1)}$	$\cdots$	
	$\vdots$							
	$q-1$	$\phi_{11}^{(q-1)}$	$\phi_{22}^{(q-1)}$	$\ldots$	$\phi_{pp}^{(q-1)}$	$\phi_{p+1,p+1}^{(q-1)}$	$\cdots$	
	q	$\phi_{11}^{(q)}$	$\phi_{22}^{(q)}$	$\ldots$	ϕ_p	0	0	$\ldots$
	$q+1$	$\phi_{11}^{(q+1)}$	$\phi_{22}^{(q+1)}$	$\ldots$	ϕ_p	$\frac{0}{0}$	$\frac{0}{0}$	$\cdots$
	$\vdots$				ϕ_p	$\frac{0}{0}$	$\frac{0}{0}$	$\ldots$

3.3.2 Parameter estimation

Once, the "order" of the model is determined, the next step is to estimate the parameters. In the ARIMA model $\phi(B)\Phi(B^s)\nabla_s^D = \theta(B)\Theta(B^s)e_t$, the parameters that need to be estimated are $\underline{\phi} = (\phi_1, \phi_2, \ldots, \phi_p)^T$, $\underline{\Phi} = (\Phi_1, \Phi_2, \ldots, \Phi_P)^T$, $\underline{\theta} = (\theta_1, \theta_2, \ldots, \theta_q)^T$ and $\underline{\Theta} = (\Theta_1, \Theta_2, \ldots, \Theta_Q)^T$. The principal method for estimating the parameters is maximum likelihood.

Let us consider $N = n+d$ observations assumed to be generated by an ARIMA(p, d, q). The unconditional likelihood is given by

$$l(\phi, \theta, \sigma) = f(\phi, \theta) - n \ln \sigma - \frac{S(\phi, \theta)}{2\sigma^2} \tag{3.33}$$

Here, the noise series, e_t, is assumed to follow a normal distribution with zero mean and variance σ^2. $f(\phi, \theta)$ is a function of ϕ and θ. The unconditional sum of squares function is given by

$$S(\phi, \theta) = \sum_{\mathbf{t=1}}^{\mathbf{n}} [\mathbf{e_t}|\mathbf{w}, \phi, \theta]^{\mathbf{2}} + [\mathbf{e}_*]'\mathbf{\Omega^{-1}}[\mathbf{e}_*] \tag{3.34}$$

Where $[e_t|\mathbf{w}, \phi, \theta] = \mathbf{E}[\mathbf{e_t}|\mathbf{w}, \phi, \theta]$ denotes the expectation of e_t conditional on $\mathbf{w}, \phi, \theta$. $\mathbf{e}_* = (\mathbf{w_{1-p}}, \ldots, \mathbf{w_0}, \mathbf{e_{1-q}}, \ldots, \mathbf{e_0})'$ denotes the vector of $p+q$ initial values of the w_t and

e_t processes needed prior to time $t = 1$, $\mathbf{\Omega\sigma_e^2} = \mathbf{cov}(\mathbf{e}_*)$ is the covariance matrix of $\mathbf{e}_*$, and $[\mathbf{e}_*] = ([\mathbf{w_{1-p}}], \ldots, [\mathbf{w_0}], [\mathbf{e_{1-q}}], \ldots, [\mathbf{e_0}])'$ denotes the vector of conditional expectations ("back forecasts") of the initial values given $\mathbf{w}, \phi, \theta$. An alternative way to represent the sum of squares is as $S(\phi, \theta) = \sum_{\mathbf{t=-\infty}}^{\mathbf{n}} [\mathbf{e_t}]^{\mathbf{2}}$.

Usually , $f(\phi, \theta)$ is important only for small n. For moderate and large value of n, (3.33) is dominated by $\frac{S(\phi,\theta)}{2\sigma^2}$. Thus, the contours of the unconditional sum of squares function in the space of the parameters $f(\phi, \theta)$ are very nearly contours of likelihood and of log-likelihood. In particular, the parameter estimates, obtained by minimizing the sum of squares (3.34) which we call (unconditional or exact) least squares estimates, will provide very close approximations to the maximum likelihood estimates.

$$S(\phi, \theta) = \sum_{j=1}^{n} e_j^2 \tag{3.35}$$

The term $f(\phi, \theta)$ is a function of coefficients ϕ and θ. This is small in comparison with the sum of squares function $S(\phi, \theta)$ when the effective number of observations, n, is large. Thus, the parameters which minimize $S(\phi, \theta)$ are usually used as close approximations to maximum likelihood estimates.

3.3.3 Diagnostic testing

The last step in building the ARIMA model is diagnostic testing. This step is useful to examine the adequacy of the model and to see if potential improvements are needed. Diagnostic tests can be done through the residual analysis. The residual (one-step prediction error) for an ARMA(p,q) process can be obtained from

$$\hat{e}_t = z_t - (\sum_{i=1}^{p} \hat{\phi}_i z_{t-i} - \sum_{i=1}^{q} \hat{\theta} \hat{e}_{t-i}) \tag{3.36}$$

If the specified model is adequate, and the appropriate orders p and q are identified, it should transform the observations to a white noise process. Thus, the residuals should behave like white noise.

One way of checking the whiteness of e_t is by checking the autocorrelation for e_t. If there is only one spike at $t = 0$ with magnitude 1, and all the other autocorrelations are equal to zero, then e_t is white noise. In the other words, if the model is appropriate, the autocorrelation should not differ significantly from zero for all lags greater than one. If the form of the model were correct and if the true parameter values are known, then the standard error of the residual autocorrelation would be $n^{-1/2}$. Any residual ACF more than two standard errors from zero would indicate that the residuals were not white, and therefore that the model orders were not accurate.

Another way to test for the whiteness of the residuals is a chi-square test of model adequacy. The test statistic is

$$Q = n \sum_{k=1}^{K} r_e^2(k) \tag{3.37}$$

which is approximately chi-square distributed with $K - p - q$ degrees of freedom if the model is appropriate. If the model is inadequate, the calculated value of Q will be inflated. Thus, we should reject the hypothesis of model adequacy if Q exceeds an approximate small upper tail point of the chi-square distribution with $K - p - q$ degrees of freedom.

As explained above, these steps are iterative steps. If the fitted model is not adequate, the identification process will continue until the fitted model is adequate.

3.3.4 Forecast

If we have completed the identification process by building the ARIMA model, now we are ready to use the model to forecast future observations. If the current time is denoted by t, the forecast for z_{t+m} is called the m-step-ahead forecast and denoted by $\hat{z}_{t+m}(t)$. The standard criterion to obtain the best forecast is to minimize the mean squared error. It can be shown that the best forecast in the mean square sense is the conditional expectation of z_{t+m} given current and previous observations:

$$\hat{z}_{t+m}(t) = E[z_{t+m}|z_t, z_{t-1}, \ldots, z_{t-N}] \tag{3.38}$$

Let us illustrate (3.38) through this example. Consider an ARIMA (1,1,1) model

$$(1 - 0.3B)(1 - B)z_t = (1 - 0.1B)e_t$$

$$(1 - 1.3B + 0.3B^2)z_t = (1 - 0.1B)e_t$$

$$z_t = 1.3z_{t-1} - 0.3z_{t-2} + e_t - 0.1e_{t-1}$$

For the one step ahead forecast, replace t by $t + 1$ then take the conditional expectation on both sides

$$z_t(1) = 1.3z_t - 0.3z_{t-1} + e_t(1) - 0.1e_t$$

$$z_t(1) = 1.3z_t - 0.3z_{t-1} - 0.1e_t$$

where $e_t(1) = E[e_{t+1}|e_t, e_{t-1}, \ldots] = E[e_{t+1}] = 0$ since e_t is white noise. Similarly, $e_t(j) = 0$ for $j > 0$. Then, for two step ahead forecasts, replace t by $t + 2$ then take the conditional expectation on both sides

$$z_t(2) = 1.3z_t(1) - 0.3z_t + e_t(2) - 0.1e_t(1)$$

$$z_t(2) = 1.3z_t(1) - 0.3z_t$$

Hence, for an m step ahead forecast, it becomes

$$z_t(m) = 1.3z_t(m-1) - 0.3z_t(m-2) \text{ for } m > 2$$

In general, for a given model, the m step ahead forecast can be found using the following procedures

1. expand the given model until an explicit expression for z_t is obtained
2. replace t by $t + m$ and then take the conditional expectation
3. apply the following properties to the equation obtained by step 2

$$
\begin{aligned}
E[z_{t+j}|z_t, z_{t-1}, \ldots] &= z_t(j) \quad \text{for } j = 1, 2, \ldots \\
E[z_{t-j}|z_t, z_{t-1}, \ldots] &= z_{t-j} \quad \text{for } j = 0, 1, 2, \ldots \\
E[e_{t+j}|e_t, e_{t-1}, \ldots] &= 0 \qquad \text{for } j = 1, 2, \ldots \\
E[e_{t-j}|e_t, e_{t-1}, \ldots] &= e_{t-j} \quad \text{for } j = 0, 1, 2, \ldots
\end{aligned}
\tag{3.39}
$$

CHAPTER 4

NEURAL NETWORKS FOR FORECASTING

The problem with the time series models is that they assume a linear relationship between the current and future values of load and a linear relationship between weather variables and load. To overcome this problem, neural networks offer the potential for general purpose nonlinear time series forecasting. As stated in [42], a good nonlinear model should be general enough to capture some of the nonlinear phenomena in the data. Neural network load forecasters can be thought of as mappings from a set of previous load, current load and future climatology variables (i.e. temperature, humidity, etc) to future load. It has been shown that several types of neural networks are universal approximators, which means that they can be used to approximate arbitrary complex mappings [43].

4.1 Neural Networks Overview

The basic neural network building block consists of the elementary computational unit or neuron, as seen in Fig. 4.1. The output of the neuron is a nonlinear function of the weighted sum of the neuron input p and bias b. The weight, w, and the bias, b are the adjustable parameters of the neuron. The nonlinear function f is called the neuron transfer function or activation function.

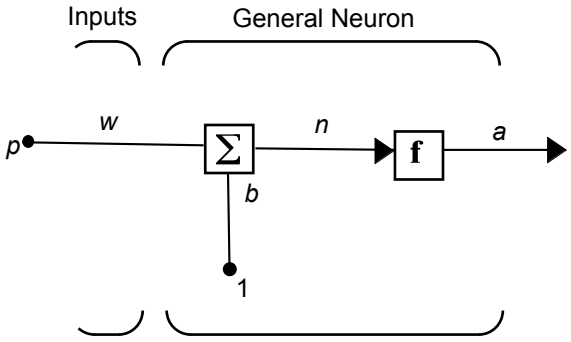


Figure 4.1: General Neuron

One neuron with a single input might not be sufficient. We need to have multiple neurons working in parallel. This set of neurons is called a layer. If we have S-neurons and R-inputs, a one layer neural network can be drawn as

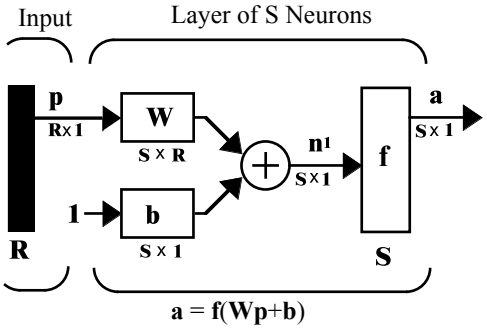


Figure 4.2: Layer of S Neurons

A single layer of neurons still does not have the capability of approximating arbitrary functions. To approximate arbitrary functions, the network needs to be extended by cascading several layers together, as shown in Fig. 4.3. Here, the output of the first layer is the input for the second layer and the output from the second layer is the input for the third layer. A layer whose output is the network output is called the output layer, and the other layers are called hidden layers. This kind of architecture is called a multilayer network. A two layer network, with sigmoid transfer functions in the first layer and linear transfer function in the second layer, is able to approximate arbitrary functions. Multilayer neural networks can be used to create general time series models [12].

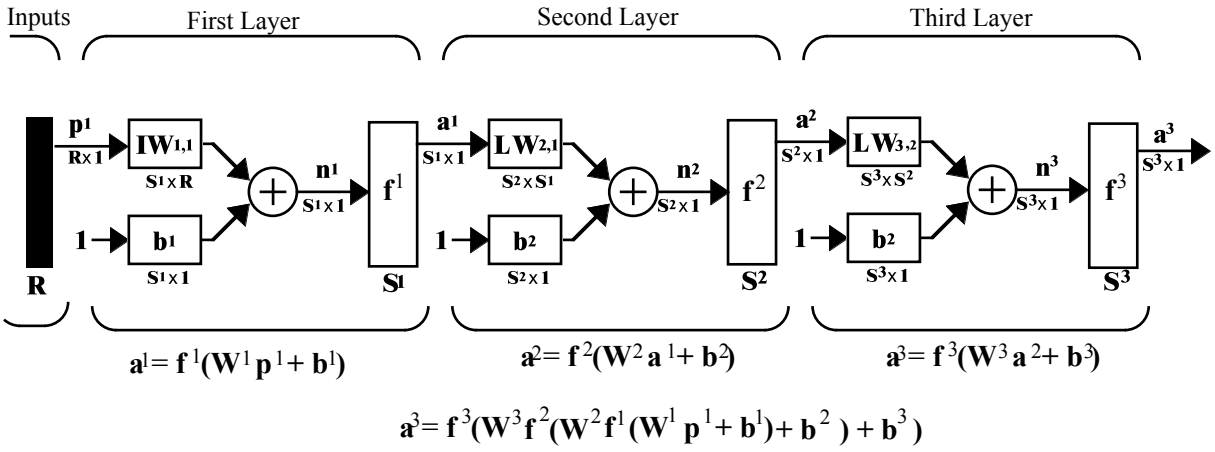


Figure 4.3: Multi Layer

Before discussing networks that can be used for forecasting, we need to introduce the tapped delay line. This is a mechanism for storing previous values of a time series, as shown in the following figure. In the diagram on the left, the thin line at the top right represents the undelayed output z_t, while the thick line represents a vector consisting of the outputs of the delay blocks.

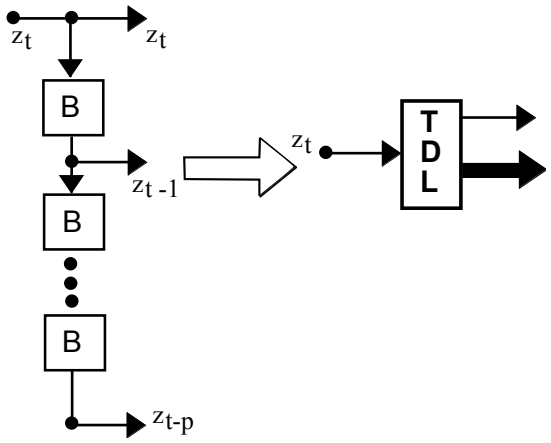


Figure 4.4: Tapped Delay Line

The tapped delay line is used in dynamic neural networks for forecasting. The NARX network (Nonlinear AutoRegressive model with eXogenous input) is a recurrent dynamic network, with feedback connections enclosing several layers of the network. The NARX network is an important and useful model for discrete-time nonlinear systems. The NARX

model is based on the linear ARX model. The defining equation for the NARX model is

$$\hat{z}_t = f(u_{t-1}, u_{t-2}, \ldots, u_{t-p}, z_{t-1}, z_{t-2}, \ldots, z_{t-r}) \tag{4.1}$$

where u_t and z_t represent the input and output of the network at time t, and p and r are the input and output order and the function f is a nonlinear function. The next value of the dependent output signal z_t is regressed on previous values of the output signal and previous values of an independent (exogenous) input signal. When the function f can be approximated by a Multilayer Perceptron (MLP), the resulting system is called a NARX neural network [44]. A diagram of the NARX network is shown below in Fig. 4.5, where a two-layer feedforward network is used for the approximation. This implementation also allows for a vector ARX model, where the input and output can be multidimensional. The

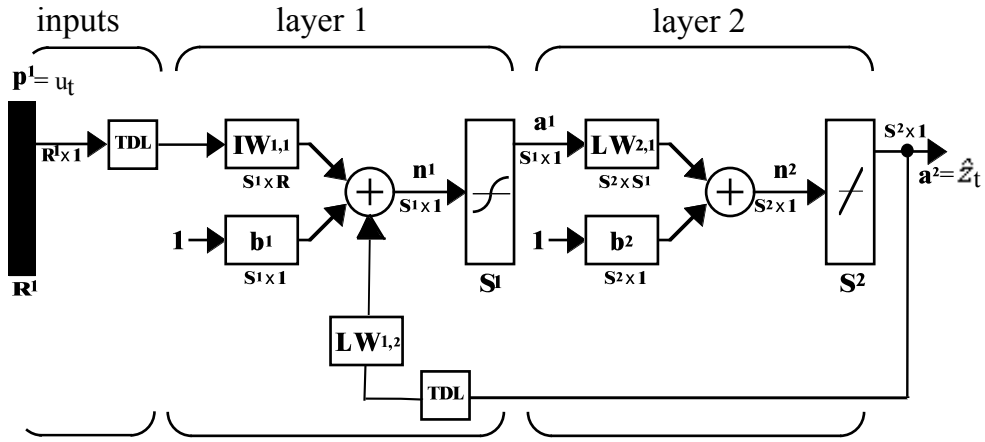


Figure 4.5: NARX Neural Network

NARX network can be trained with static backpropagation. The two tapped delay lines can be replaced with extended vectors of delayed inputs and targets. The NARX network can be implemented in two ways, which are shown in Fig. 4.6. The first architecture is called the parallel architecture, in which the predicted output is fed back to the input of the feedforward neural network, as shown in the left figure below. The second architecture is the Series-Parallel architecture. This architecture uses target outputs in place of the predicted outputs in the feedback loop, hence simple training algorithms, such as backpropagation, can be easily implemented.

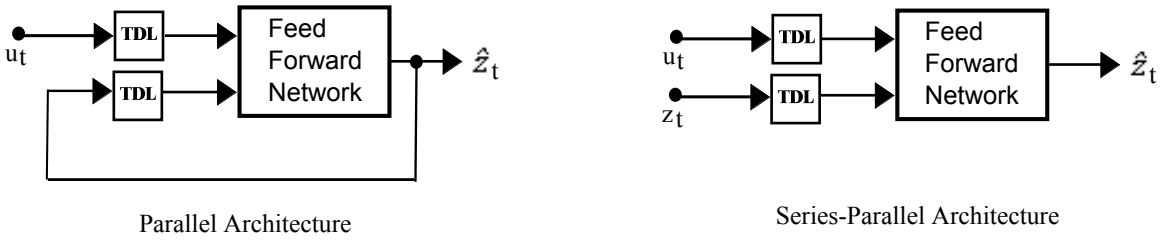


Figure 4.6: NARX Neural Network Architecture

By using neural networks, the ARMA model of Eq. (3.19) can be generalized to non-linear systems. To generalize the ARMA model, first re-write it in the following form

$$z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + \ldots + \phi_p z_{t-p} - [\theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q}] + e_t \qquad (4.2)$$

This can also be written as

$$z_t = \hat{z}_t + e_t \qquad (4.3)$$

where $\hat{z}_t$ is a forecast of z_t. This system can be represented by the following block diagram.

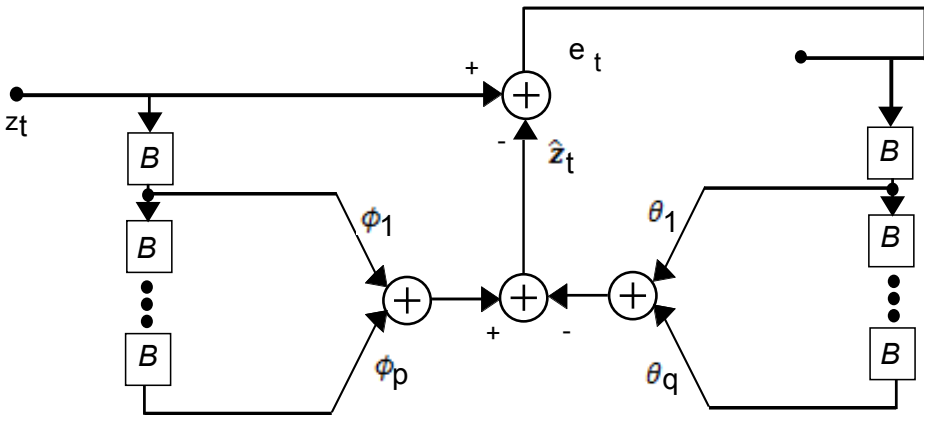


Figure 4.7: ARMA Model

This system can be represented in abbreviated notation, as the following figure.

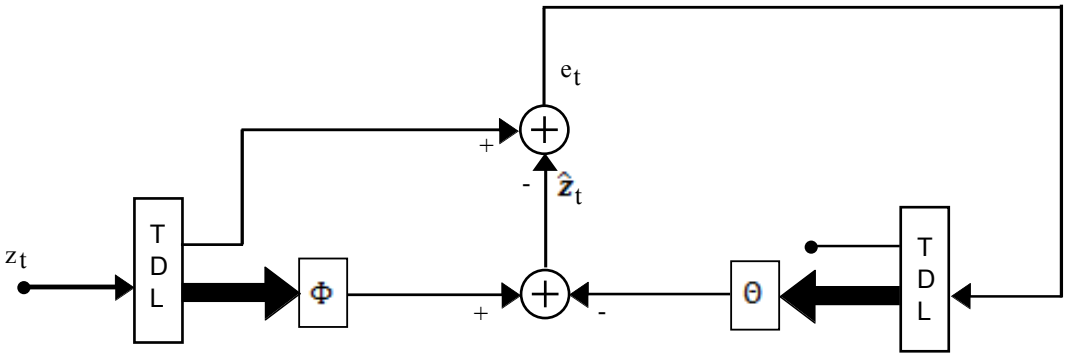


Figure 4.8: Abbreviated Notation of the ARMA Model

By combining the tapped delay line with a multilayered neural network, a nonlinear version of the ARMA model (NARMA) can be created, as shown in the following figure.

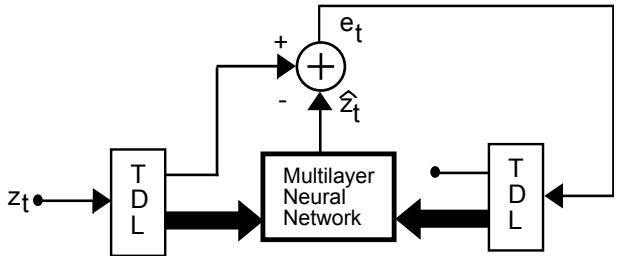


Figure 4.9: Nonlinear ARMA (NARMA) Predictor Model Using a Neural Network

Here, the previous values of the time series (z_t) are combined in a nonlinear way with current and previous values of the forecasting errors (e_t) to form a forecast of future values of the time series. Compare this with the linear ARMA equations (3.19)

$$\hat{z}_t = f(z_{t-1}, z_{t-2}, \ldots, z_{t-p}, e_{t-1}, e_{t-2}, , e_{t-r}) \tag{4.4}$$

To generalize the periodic model given in Eq. 3.24 and 3.25, the periodic tapped delayed line is needed. The generalization of the periodic ARIMA model is the periodic NARIMA (PNARIMA) model shown in the following figure

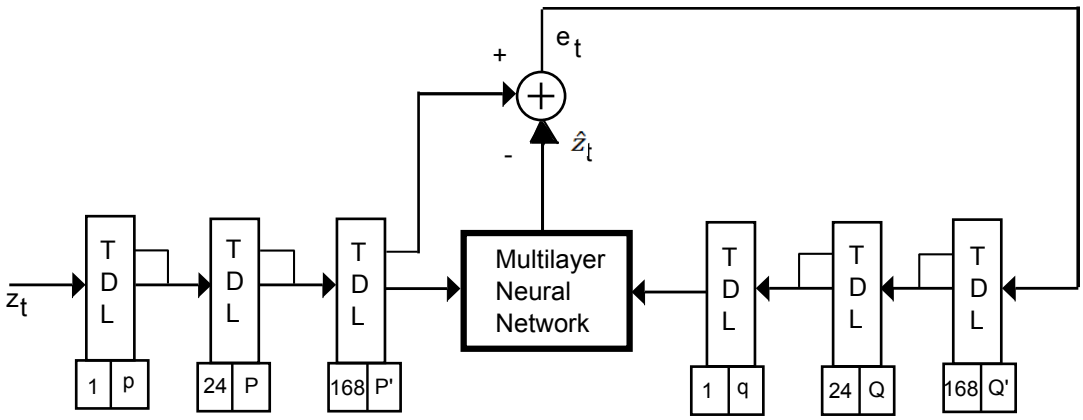


Figure 4.10: Periodic NARIMA (PNARIMA) Model

4.2 Designing neural networks for forecasting

The task of designing neural networks for forecasting is an iterative procedure which begins by collecting data and pre-processing them to make training more efficient. Then, training data need to be divided into training, validation and testing sets. After that, the appropriate network type and architecture for forecasting are chosen. Once the network and architecture have been decided, the next step is to select a training algorithm that is appropriate for the forecasting problem. After training the network, we need to analyze the network in order to see whether the performance is satisfactory. If we find any problem, we have to re-start our process from the beginning, as shown in Fig. 4.11

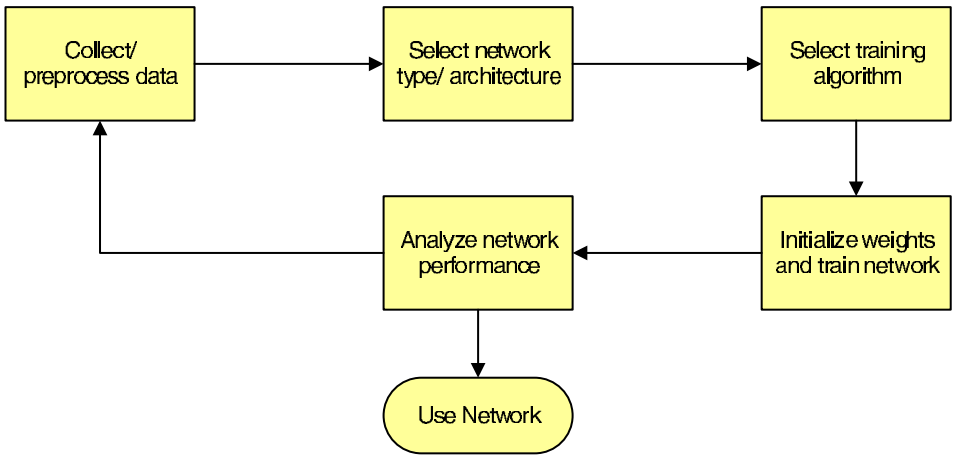


Figure 4.11: Flowchart of Neural Network Training Process [43]

4.2.1 Data Collection and pre-processing

The first stage in designing the neural network for forecasting is collecting data. The required amount of data depends on the complexity of the underlying function that we are trying to approximate. The choice of data set is closely related to the choice of the number of neurons in the neural network. To make the forecasting problem more manageable, data are pre-processed before being used to train the neural network. There are several types of data pre-processing, such as normalization, nonlinear transformations, feature extraction, coding of discrete inputs/targets, handling the missing data, etc.

Normalization is the main step in data pre-processing. It will help the neural network to extract relevant information in the training process. Generally, there are two methods for normalization. The first method is to normalize the data so that they fall into a standard range - typically -1 to 1. This can be done by

$$\mathbf{p}^n = 2(\mathbf{p} - \mathbf{p}^{min})./(\mathbf{p}^{max} - \mathbf{p}^{min}) - 1 \tag{4.5}$$

where $\mathbf{p^{min}}$ is the vector containing the minimum values of each element of the input vectors in the data set, $\mathbf{p^{max}}$ contains the maximum value. The other method of normalization is to adjust the data so that they have specified mean and variance - typically 0 and 1:

$$\mathbf{p}^n = (\mathbf{p} - \mathbf{p}^{mean})./\mathbf{p}^{std} \tag{4.6}$$

where $\mathbf{p^{mean}}$ is the average of the input vectors in the data set and $\mathbf{p^{std}}$ is the vector containing the standard deviations of each element of the input vector.

Another type of pre-processing is removing outliers, missing values or any irregularities. Neural networks can be sensitive to defective data. It is not possible to guarantee network performance when the inputs to the networks are outside the range of the training set. Missing data can occur in the inputs and/ or targets (e.g., historical load data is not available for one month). If missing data are in an input variable, one possibility is to replace the missing data with the average value for the particular input variable. When

missing data occur in an element of the target, then the performance index can be modified so that errors associated with the missing target values will not contribute to the performance index.

Once data collection and pre-processing have been implemented, the data will be divided into three sets: training, validation and testing. The training set generally makes up approximately 70% of the full data set, with validation and testing making up approximately 15% each. The gradient is computed on the training set, and the mean square error on the training set is minimized. During training, validation error is monitored. Training is stopped when validation error increases, to prevent overfitting. After training is complete, the error on the test set is computed. This provides an indication of how the network will perform on new data.

4.2.2 Selecting the network type and architecture

The purpose of forecasting is to predict the future value of some time series, e.g., short term load forecasting predicts the load up to 24 hours ahead. Since many forecasts require nonlinear models, dynamic neural networks can be used. In this research, we will be using the PNARIMA model shown in Fig. 4.10. For this network, we will need to select the number of neurons in the hidden layers and the number of hidden layers. To determine the number of hidden layers, the standard procedure is to begin with a network with one hidden layer. If the performance of the two-layer network is not good enough, then a three-layer network can be used. It would be unusual to use more than two hidden layers, because if there are multiple hidden layers the training becomes more difficult. Each layer in the hidden layer performs a squashing operation, as the activation function in the hidden layer must be differentiable and nondecreasing i.e. logistic or hyperbolic tangent. This causes the derivatives of the performance with respect to weights in the early layers to be quite small, which can cause slow convergence for steepest descent optimization. It has been shown that one or two hidden layer is good enough for neural network forecasting [3].

We also need to select the number of neurons in each layer. Choosing the number of neurons in the hidden layers is not straightforward. If they are too few, the model will not be flexible enough to model the data well. On the other hand, if they are too many, the model will overfit the data. The standard procedure in choosing the number of neurons in the hidden layers is to begin with more neurons than necessary, which may cause the network to overfit the data. To prevent the overfitting, we can use early stopping or Bayesian regularization. In Bayesian regularization, the norm of the weights is penalized, which reduces the effective number of parameters. If, after training, the effective number of parameters is much less than the total number of parameters, then the number of neurons can be reduced and the network retrained. It is also possible to use ”pruning” methods to eliminate neurons or weights in the network.

4.2.3 Selecting training algorithm

There are several processes in training; choosing the performance function, initializing the weights, choosing the training algorithm, and choosing the criterion for stopping training.

The type of weight initialization will depend on the type of network. In general, the weights and biases are set as small random values (e.g., uniformly distributed between -0.5 and 0.5, if the inputs are normalized to fall between -1 and 1) for multilayer networks. We have to avoid setting the weights and biases to zero, because then the initial condition may fall on a saddle point of the performance surface. On the other hand, if we make the initial weights too large, the initial condition can fall on flat part of the performance surface, because of the saturation of the sigmoid transfer functions. There is another good approach to setting the initial weights and biases for a two-layer network. It was introduced by Widrow and Nguyen [45]. This method generates initial weights and bias values for a layer, so that the active regions of the neurons will be distributed approximately evenly over the input space. In the other words, it is to set the magnitude of weights in the first layer so that the linear region of each sigmoid function covers $1/S^1$ of the range of the input.

The biases are set randomly in the region $-1 < x < 1$ so that the center of each sigmoid function falls randomly in the input space by assuming the inputs to the network have been normalized to values between -1 and 1. The detail is as follows, first set row i of $\mathbf{W^1}$, ${}_i\mathbf{w}$, to have a random direction and a magnitude of

$$\|{}_i w\| = 0.7(S^1)^{1/R}$$

Set b_i to a uniform random value between $-\|{}_i w\|$ and $\|{}_i w\|$.

After initializing the weights, the weights and biases of the network are adjusted so as to minimize the error, e, between the network output and the proper function response, as shown in Fig. 4.12.

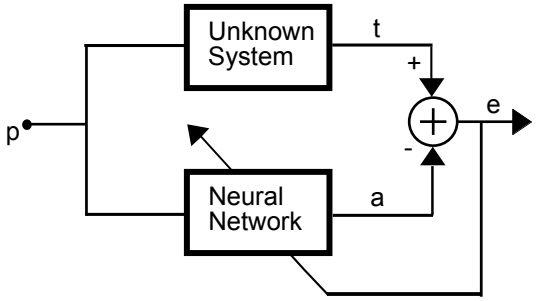


Figure 4.12: Neural Networks

The training process can be thought of as a standard optimization problem, in which the objective function is the mean squared network errors:

$$F(\mathbf{x}) = \frac{1}{QS^M} \sum_{q=1}^{Q} (\mathbf{t}_q - \mathbf{a}_q)^T (\mathbf{t}_q - \mathbf{a}_q) \tag{4.7}$$

or

$$F(\mathbf{x}) = \frac{1}{QS^M} \sum_{q=1}^{Q} \sum_{i=1}^{S^M} (t_{i,q} - a_{i,q})^2 \tag{4.8}$$

where $\mathbf{a_q}$ is the neural network output for the qth input, $\mathbf{p_q}$, and M is the number of layer.

Since training can be considered as an optimization problem, there are many different optimization algorithms that can be used. The Levenberg-Marquardt algorithm and some variations of the conjugate gradient algorithm appear to produce the best results [43]. We

will use the Levenberg-Marquardt algorithm. Levenberg-Marquardt blends the gradient descent method with the Gauss-Newton method. This algorithm inherits the speed advantage of the Gauss-Newton algorithm and the stability of steepest descent. The training process can be summarized as follows

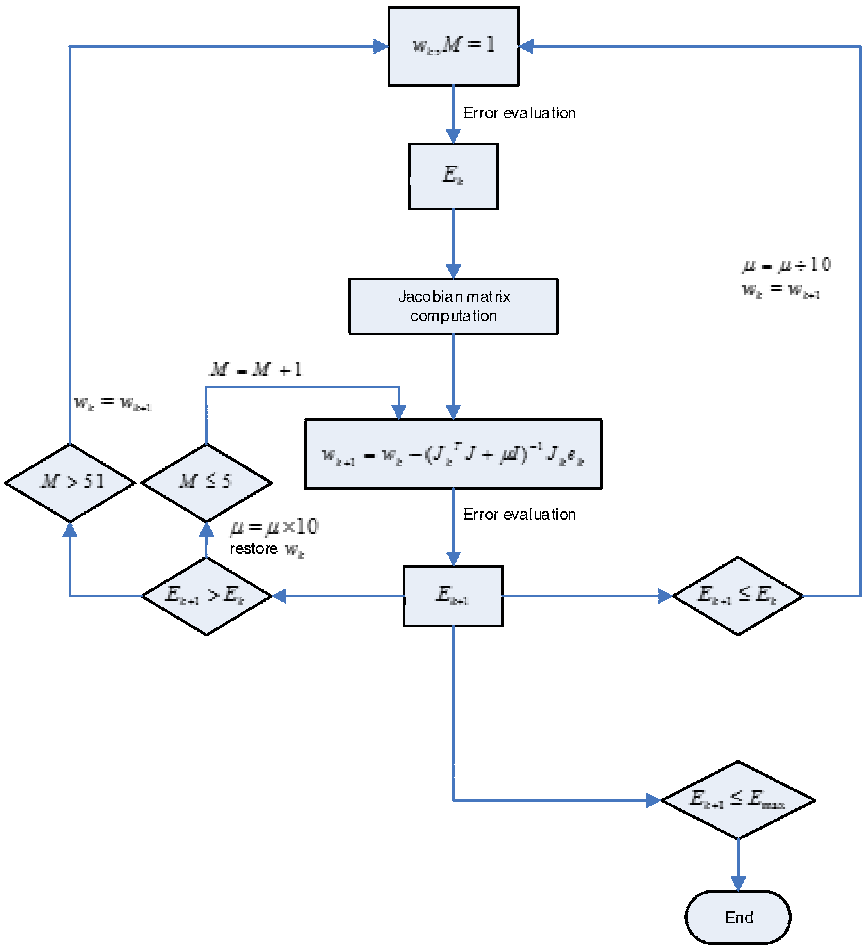


Figure 4.13: Levenberg-Marquardt Flowchart

1. Initialize the weight (randomly generated), then evaluate the performance

2. Update the weights

$$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k e_k \tag{4.9}$$

   where J is the Jacobian matrix

3. Calculate the performance using the updated weight

4. If the performance increases after updating the weight, then use the previous weight and expand the coefficient μ by a factor of 10. Go to step 2 and try an update again

5. If the performance decreases after updating the weight, then accept the updated weight as the current one and contract the coefficient μ by 10

6. Go to step 2 with the new weights until the performance is smaller than the required value

As mentioned above, the main goal of training is to get a network that has small errors on the training set, while still responding properly to novel inputs. A network is said to generalize well when a network can perform as well as on novel inputs as on training set inputs. The network will continue to work well under all operating conditions confidently. One technique to improve the generalization is to divide the data into three subsets: training, validation and test. The validation data is used to stop the training early when the network begins to overfit [46]. Another method that is very helpful in improving network generalization is Bayesian regularization [47]. This method constrains the network weights and biases so that the network response will be smooth. This will guarantee that the network will interpolate reasonably between points in the training set. In this algorithm, a penalty term is added into the original objective function:

$$F(\mathbf{x}) = \beta E_D + \alpha E_W = \beta \sum_{q=1}^{Q} (\mathbf{t}_q - \mathbf{a}_q)^T (\mathbf{t}_q - \mathbf{a}_q) + \alpha \sum_{i=1}^{n} x_i^2 \tag{4.10}$$

where E_W is the sum of squares of the network weights, and α and β are objective function parameters. If $\alpha << \beta$, then the training algorithm will drive the errors smaller. If $\alpha >> \beta$, training emphasizes weight reduction at the cost of network errors, thus producing a smoother network response. The steps of this algorithms, based on [47], can be described below

1. Initialize α and β and the weights. It is suggested to set $\alpha = 0$ and $\beta = 1$ and to use the Nguyen-Widrow method for weight initialization. Compute E_D and E_W using

the initialization parameters with $\gamma = n$

2. Take one step of the Levenberg-Marquardt algorithm to minimize the objective function $F(\mathbf{x}) = \beta E_D + \alpha E_W$

3. Compute the effective number of parameter $\gamma = N - 2\alpha \text{tr}(\mathbf{H})^{-1}$ making the Gauss-Newton approximation to the Hessian available in the Levenberg-Marquardt training algorithm

$$\mathbf{H} = \nabla^2 F(\mathbf{w}) \approx 2\beta \mathbf{J}^T \mathbf{J} + 2\alpha \mathbf{I}_N$$

where **J** is the Jacobian matrix of the training set errors

4. Compute new estimates for the objective function parameter

$$\alpha = \frac{\gamma}{2E_W(\mathbf{w})}$$

and

$$\beta = \frac{n - \gamma}{2E_D(\mathbf{w})}$$

5. Iterate step 2 through 4 until convergence

After choosing the training algorithm, we need some criteria to stop the training. The network has to stop the training when the error reaches some specified limits. The simplest criterion is to stop the training after a fixed number of iterations have been reached. But since it is also difficult to know how many iterations will be required, the maximum iteration number is generally set reasonably high. Another stopping criterion is through the norm of the gradient of the performance index. If this norm reaches a sufficiently small threshold, then the training can be stopped. Since the gradient should be zero at a minimum of the performance index, this criterion will stop the algorithm when it gets close to the minimum. The threshold for the minimum norm should be set to a very small value (e.g., 10^{-6} for mean square error indices, with normalized targets) so that the training does

not end prematurely. When early stopping is used, the training will be stopped when the performance on the validation set increases for a set number of iterations. In addition to preventing overfitting, this stopping procedure also provides a significant reduction in computation; for most practical problems, the validation error will increase before any of the other stopping criteria are reached.

4.2.4 Analyzing network performance

In time series forecasting, we also need to test the performance of the network after training is complete. There are two important concepts that are used when analyzing a trained forecasting network

1. the forecasting error should not be correlated in time and
2. the forecasting error should not be correlated with the input sequence

If the forecasting errors are correlated in time, we need to improve our original prediction. In order to test the correlation of the forecasting errors in time, we can use the sample autocorrelation

$$R_e(\tau) = \frac{1}{Q-\tau}\sum_{t=1}^{Q-\tau} e(t)e(t+\tau) \tag{4.11}$$

If the forecasting errors are uncorrelated (white noise), then we would expect $R_e(\tau)$ to be close to zero, except when $\tau = 0$. To determine if $R_e(\tau)$ is close to zero, we can set an approximate 95% confidence interval using the range

$$-\frac{2R_e(0)}{\sqrt{Q}} < R_e(\tau) < \frac{2R_e(0)}{\sqrt{Q}} \tag{4.12}$$

The error $e(t)$ is white, if $R_e(\tau)$ satisfies (4.12) for $\tau \neq 0$. To test the correlation between the forecasting errors and the input sequence, we can use the sample cross correlation function

$$R_e(\tau) = \frac{1}{Q-\tau}\sum_{t=1}^{Q-\tau} p(t)e(t+\tau) \tag{4.13}$$

If there is no correlation between the forecasting errors and the input sequence, then we would expect $R_{pe}(\tau)$ to be close to zero for all τ. To determine if $R_{pe}(\tau)$ is close to zero, we can set an approximate 95% confidence interval using the range

$$-\frac{2\sqrt{R_e(0)}\sqrt{R_p(0)}}{\sqrt{Q}} < R_{pe}(\tau) < \frac{2\sqrt{R_e(0)}\sqrt{R_p(0)}}{\sqrt{Q}} \tag{4.14}$$

In summary, a neural network can be said to be properly trained if [11]

1. it is well fitted to the training data
2. its performances on the training sample and on the test samples are comparable
3. its performances across different test samples are coherent.

CHAPTER 5

RESULTS

5.1 Data Description

For this study, data were obtained from PT. Perusahaan Listrik Negara (State Electricity Company) Batam. Batam is an industrial city in Indonesia. This region has become a free trade zone since 1989 as part of the Sijori Growth Triangle. Shipbuilding and electronics manufacturing are important industries in this area.

As an Indonesian government-owned corporation, which has a monopoly on electricity distribution in Indonesia, PT. PLN Batam must meet public electricity demands at all times. Moreover, the economic growth in Batam has become very rapid. PT. PLN Batam has to be more diligent in delivering electricity to industrial and residential consumers.

In distributing their services, PT. PLN Batam divides the consumers into 5 groups; households, business, industry, multipurpose, and general consumers. Household is defined as residential, individual or social organisations who use the electricity personally and for daily activities. The business consumer is a commercial organization or small industry, such as hotels, banks, law firms, etc. Industrial consumers are large-scale industries, e.g., manufacturing. The general consumer is a non-profit entity, such as schools, hospitals or religious organizations. The last category is the multipurpose consumer. Government buildings, street lights, or Base Transceiver Stations (BTS) can be classified as multipurpose consumers. As of 2010, the total number of customers, based on data from PT. PLN Batam, can be seen in Table 5.1.

In operation, PT. PLN Batam has a mixed fuel strategy. As of 2011, PT. PLN Batam generated energy from a mixture of 95.19% gas and 4.81% other fossil fuel. In the same

Table 5.1: Classification and Composition of Consumers [48]

Year	Households	Business	Industry	General	Multipurpose	Total
2004	109112	14278	177	1571	1478	126616
2005	123692	15258	181	1802	1380	142313
2006	138095	16437	203	2025	1281	158041
2007	151025	18191	232	2222	1157	172827
2008	164776	19258	260	2411	1524	188229
2009	178888	20774	279	2892	1064	203897
2010	187116	22367	276	3003	1045	213807

year, the total installed capacity was 373 MW.

For the purpose of this study, the data is hourly electricity consumption data, which was recorded over a 2 year period from May 2009 until April 2011. Fig. 5.1 shows typical behaviour of hourly electric consumption for August 2009 for a week. Typically, peak load occurs at 10 AM, 2 PM and 7 PM on week days. The lowest electricity consumption is typically at 7 AM. It is presumed that at 7 AM consumers turn the lights off, and then at 7 PM they come back from work and perform activities at home that use a large amount of electricity. The electricity consumption is also lower during the weekend (Saturday and Sunday) or on the holidays. For example, on August 17, 2009 the lowest electricity consumption was 105.7 MW at 8 AM and the largest was 157 MW at 7 PM. These are lower than usual, because it is a holiday.

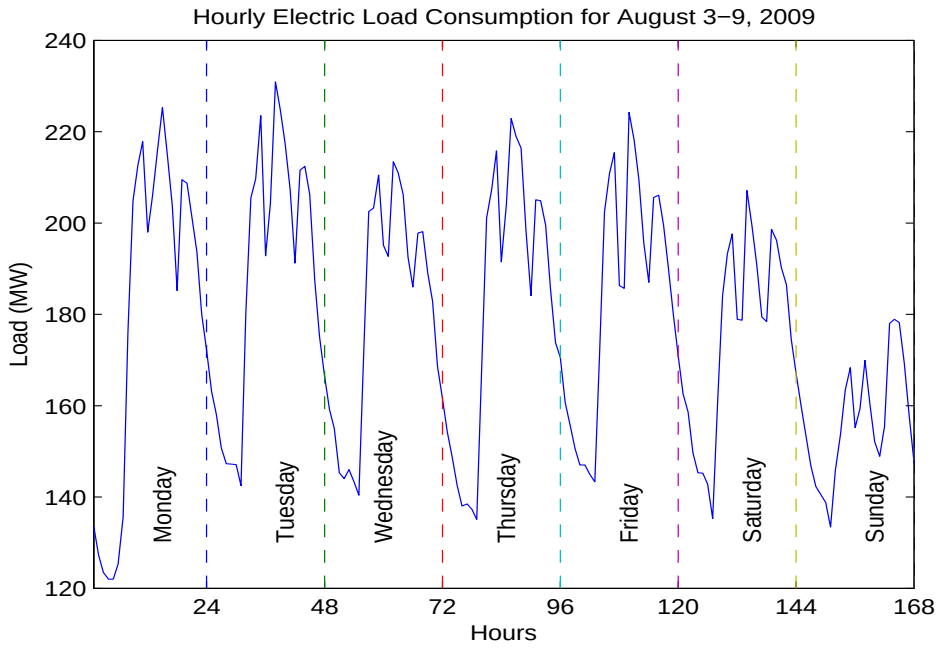


Figure 5.1: Hourly Electric Load Consumption for August 3 - 9, 2009

A plot of peak hourly load for each month from May 2009 to April 2011 can be seen in Fig. 5.2.

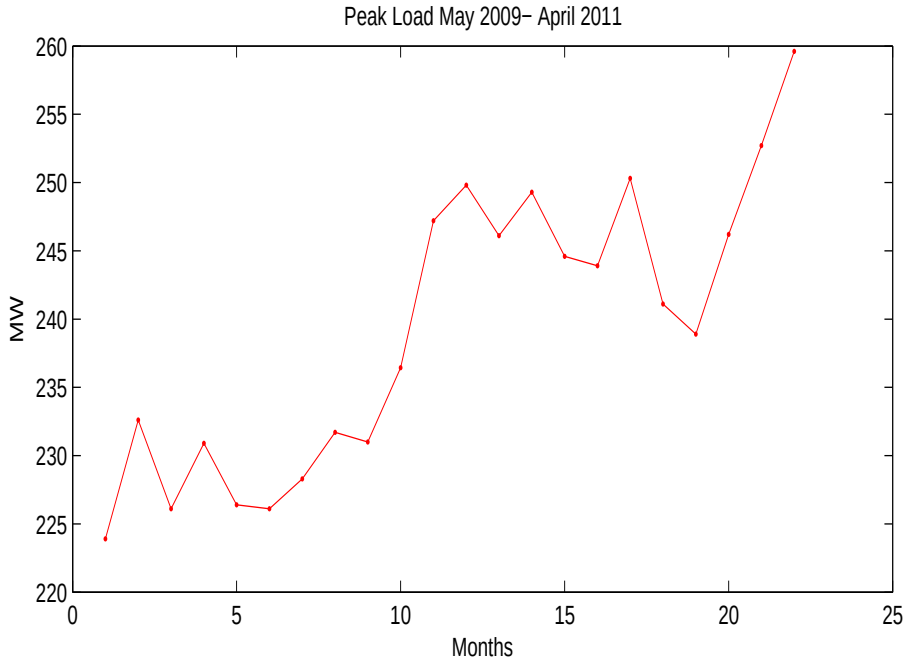


Figure 5.2: Peak Load for Each Months from May 2009 until April 2011

For this study, the only input variable that will be used for the load predictor is hourly electricity load. Weather variables (temperature, humidity, etc.) can also be considered as potential inputs for the predictor. But since Indonesia has a tropical rain forest climate with two major seasons - dry (May to October) and wet (November to April) - temperature does not change too much throughout the year. Hence, the weather variables provide no additional improvements to the predictor. Fig. 5.3 shows the relationship between the temperature and daily peak load for August 2009.

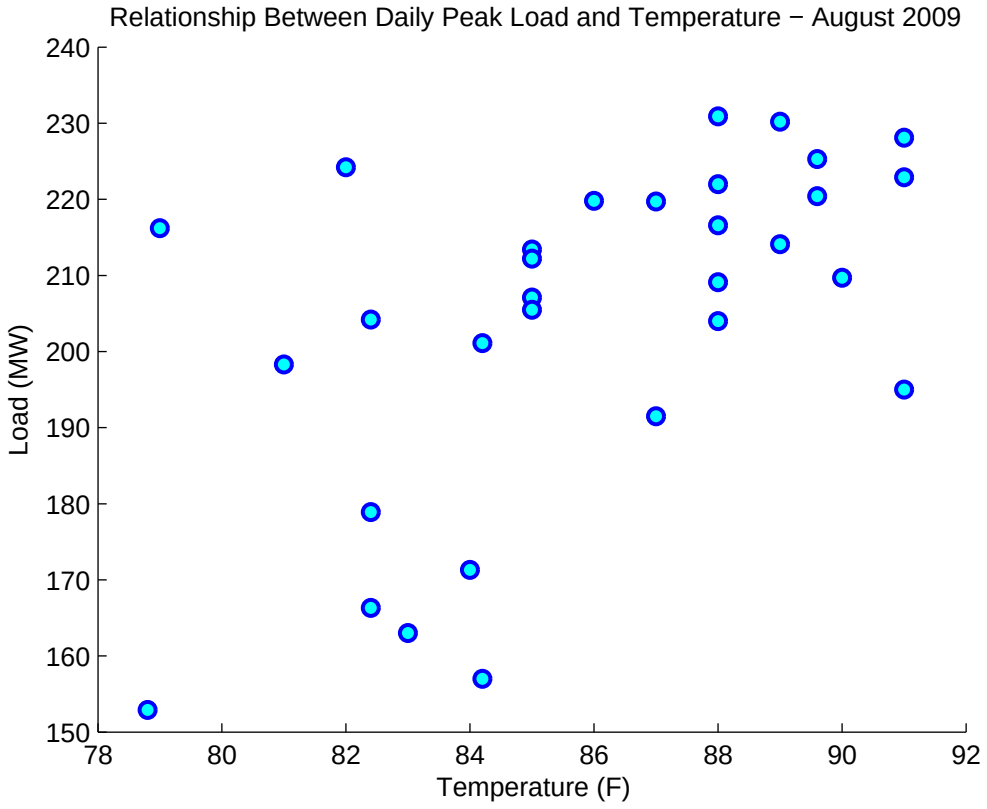


Figure 5.3: Relationship Between Temperature and Daily Peak Electric Load Consumption for August, 2009

From Fig. 5.3, we can see that temperature does not have a significant impact on the load. As stated in Fidalgo et.al [15], whether weather-dependent factors have a significant influence on the load will depend on the region and climate conditions. Temperature will not be included in our forecasting models, because it did not improve the forecasts.

In our tests, data are divided into two groups; training data and testing data. The training data are used to train ARIMA models and PNARIMA neural networks in order to estimate the model parameters. The testing data are used to test the fitted models. The purpose of the testing data is to validate the model and to predict future model performance. Training data consist of 7 data sets, and there are also 7 testing data sets. The training and testing data will assist us in determining how many different models are required throughout the year. One training set will consist of an entire year of data - meaning that one model will be fit for the entire year. We will also consider fitting two models - one for the wet season and one for the dry season. This means there will be two training sets of six months each. Finally, we will consider four models throughout the year. In this case the data will be divided into three month intervals. The following table shows the training and testing intervals.

Table 5.2: Training and Testing Periods

Models		Training Periods	Testing Periods
Full year		May, 09 - April, 10	May, 10 - April, 11
6 month	Dry	May, 09 - Oct, 09	May, 10 - Oct, 10
	Wet	Nov, 09 - April, 10	Nov, 10 - April, 11
3 month	Dry 1	May, 09 - July, 09	May, 10 - July, 10
	Dry 2	Aug, 09 - Oct, 09	Aug, 10 - Oct, 10
	Wet 1	Nov, 09 - Jan, 10	Nov, 10 - Jan, 11
	Wet 2	Feb, 10 - April, 10	Feb, 11 - April, 11

The idea will be to use the fewest models that can accurately predict the load. If the single model, fit to an entire year of data, provides predictions that are as good as the predictions made by the two six-month models, or the four three-month models, then we will recommend the full year model. The fewer models that we need to use, the more data we can use to fit the models. This should provide more accurate models. However, if the

characteristics of the load curve change significantly from one season to the next, then we will need to fit more models in order to get accurate forecasts.

Once the data have been divided into training and testing set, we are ready to train and test the models. In the following sections we will describe the following steps:

1. Fit the ARIMA model to the training data
2. Fit the PNARIMA neural network model to the training data
3. Compare the accuracy of seasonal ARIMA and PNARIMA neural network models on the testing data

In comparing the accuracy of each model, this study will use the following forecasting accuracy criteria:

1. Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(z_t - \hat{z}_t)^2}{n}} \tag{5.1}$$

2. Mean Absolute Error (MAE)

$$MAE = \frac{\sum_{t=1}^{n}|z_t - \hat{z}_t|}{n} \tag{5.2}$$

5.2 Fitting the ARIMA model

To make a linear prediction using seasonal ARIMA models, we will follow the Box-Jenkins procedures described in Chapter 3. The initial training data is the hourly electricity consumption in the 3 month period August 2009 to October 2009 (Dry 2). In the preliminary identification stage, we examine the autocorrelation function (ACF) and partial autocorrelation function (PACF) for the training data to determine whether the original process is stationary or not.

5.2.1 Preliminary model structure determination

As shown in Fig. 5.4, the slowly decaying ACF indicates that the process is not stationary. To obtain a stationary process, we need to difference the original process and identify the appropriate degree of differencing, d. There are several possible differencing schemes (hourly, daily, weekly and combinations), as shown in Fig. 5.5. The differenced process, w_t, should have an ACF that decays reasonably quickly.

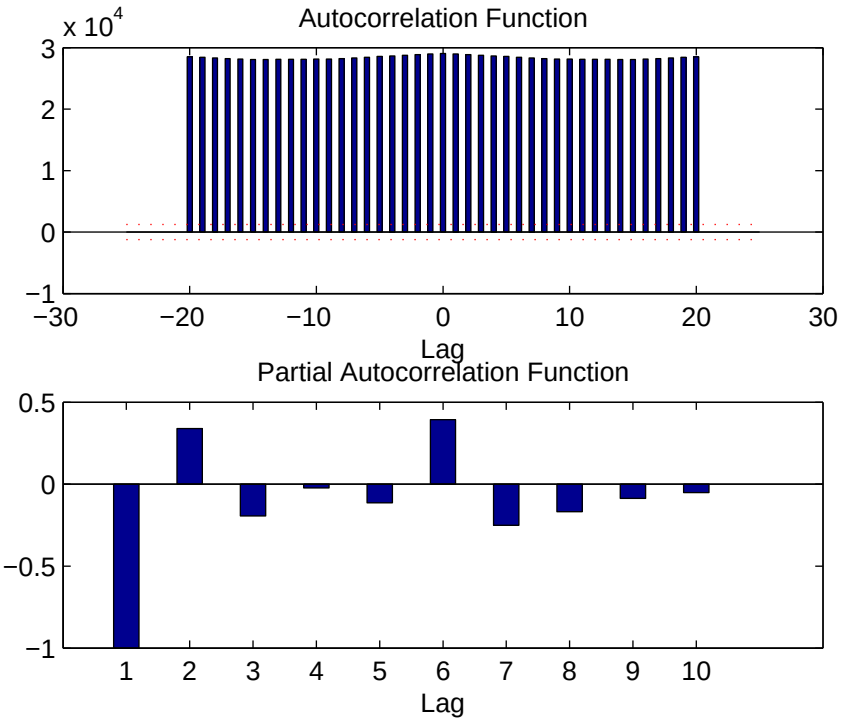
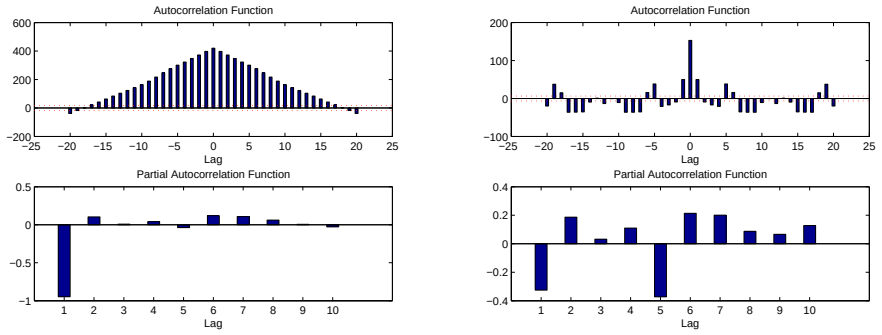


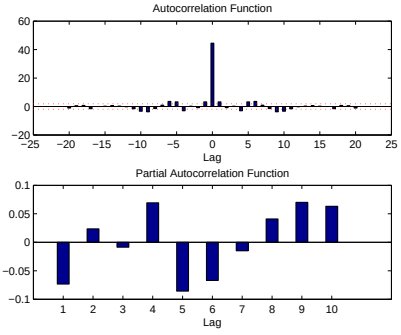
Figure 5.4: ACF and PACF from Original Load August to October, 2009

The ACF plots in Fig. 5.5 indicate that a stationary process may be obtained by either a differencing scheme of hourly and daily $\nabla_1\nabla_{24}$ or hourly and weekly $\nabla_1\nabla_{168}$. In the following discussion, we use $\nabla_1\nabla_{24}$. The autocorrelation function(ACF) and partial autocorrelation function (PACF) of the differenced time series can be seen in the Fig. 5.6.

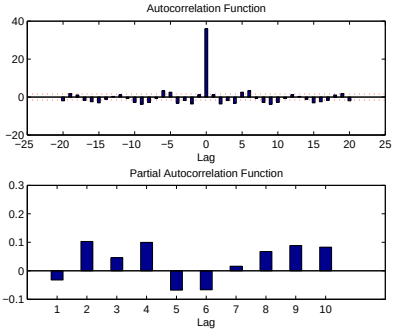


(a) daily

(b) hourly



(c) hourly and daily



(d) hourly and weekly

Figure 5.5: ACF Differenced of Load August to October, 2009

The next step in developing the model is to identify the order of the model, given by $(p, d, q) \times (P, D, Q)_{24}$. To get these model orders, we can inspect the ACF and PACF separately, as shown in Fig. 5.6. The exponential decaying pattern in the ACF and the large spikes at lag 24 in the PACF suggest a daily order, $(P, D, Q)_{24}$, of $P = 1$ and $Q = 1$. Hence, our tentative model has the form

$$\begin{aligned} \phi_p(B^1)\Phi_P(B^{24})\nabla_1\nabla_{24}z_t &= \theta_q(B^1)\Theta_Q(B^{24})e_t \\ (1 - \Phi_1 B^{24})\nabla_1\nabla_{24}z_t &= (1 - \Theta_1 B^{24})e_t \end{aligned} \tag{5.3}$$

which we can indicate by $(0, 1, 0) \times (1, 1, 1)_{24}$

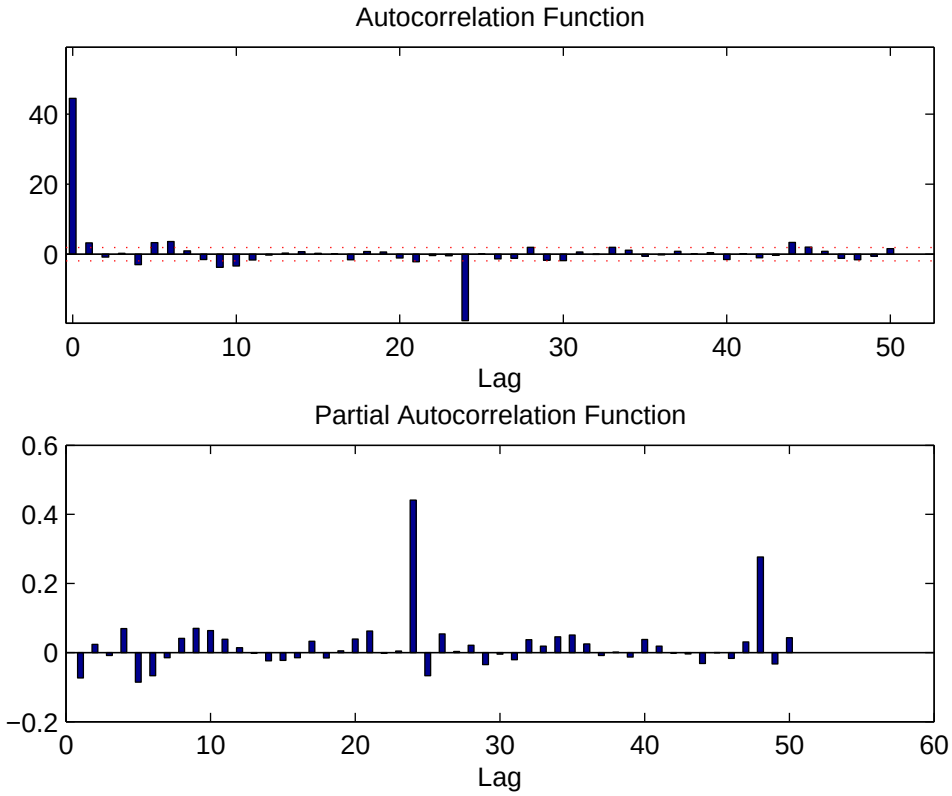


Figure 5.6: ACF and PACF Load August - October 2009 Difference $\nabla_1 \nabla_{24}$

The parameters in the initial model (5.3) were estimated using the maximum likelihood method. After estimating the parameters, we can diagnose the residuals of the model, e_t, to check the goodness of fit. If the model is accurate, the autocorrelation function of the residuals should be close to an impulse function, which would indicate that the residuals are white. On the other hand, if the model is a poor fit, the residuals will not be white.

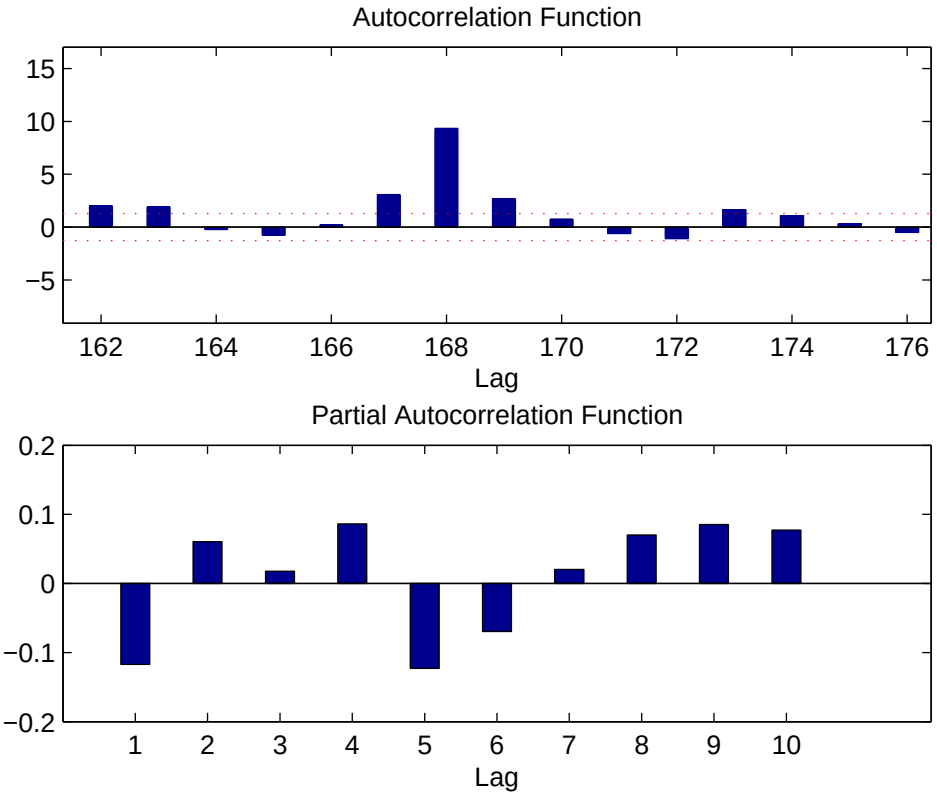


Figure 5.7: ACF and PACF Residual for Tentative Model

Fig. 5.7 shows the residual ACF for our tentative model for lags 162 to 176. Notice the dashed red lines, which are confidence intervals around zero (to see the detail of its pattern, the ACF for small lags is omitted). Most of the values are within these lines, or close by. But there is a large spike at lag 168 (recall that 168 hours is one week). This spike at lag 168 is outside the confidence interval. It shows that the model may need a weekly component. By adding this weekly component, we get the following form: $(0,1,0) \times (1,1,1)_{24} \times (1,0,1)_{168}$. Hence, our final ARIMA model is as follows

$$
\begin{aligned}
(1-\Phi_1 B^{24})(1-\Phi_1{}' B^{168})\nabla_1\nabla_{24} z_t &= (1-\Theta_1 B^{24})(1-\Theta_1{}' B^{168})e_t \\
(1-\Phi_1 B^{24}-\Phi_1{}' B^{168}+\Phi_1\Phi_1{}' B^{192})\nabla_1\nabla_{24} z_t &= (1-\Theta_1 B^{24}-\Theta_1{}' B^{168} \\
&\quad +\Theta_1\Theta_1{}' B^{192})e_t
\end{aligned}
\tag{5.4}
$$

5.2.2 Parameter estimation and model validation

The final fitted ARIMA model for the training data set August to October 2009 is

$$(1 + 0.952B^{24})(1 + 0.809B^{168})\nabla_1\nabla_{24}z_t = (1 + 0.092B^{24})(1 + 0.981B^{168})e_t \quad (5.5)$$

A whiteness test for e_t from the model above gives no indication of model inadequacy. The ACF for e_t is shown in Fig. 5.8. It shows that e_t is white. Besides the whiteness test, we can see the model adequacy from the confidence intervals. Confidence intervals can determine whether the estimated parameter is significantly different from 0. From the final fitted ARIMA parameters, we have the confidence interval for each parameters shown in Table 5.3. None of the confidence intervals include zero. We can conclude that our final model is adequate.

Table 5.3: Confidence Intervals for Final ARIMA Model Training Data August to October 2009

Parameters	Confidence Interval	
	Lower Bound	Upper Bound
Φ_1	-0.856	-0.906
Φ_1‘	-0.724	-0.850
Θ_1	-0.090	-0.191
Θ_1‘	-0.931	-1.009

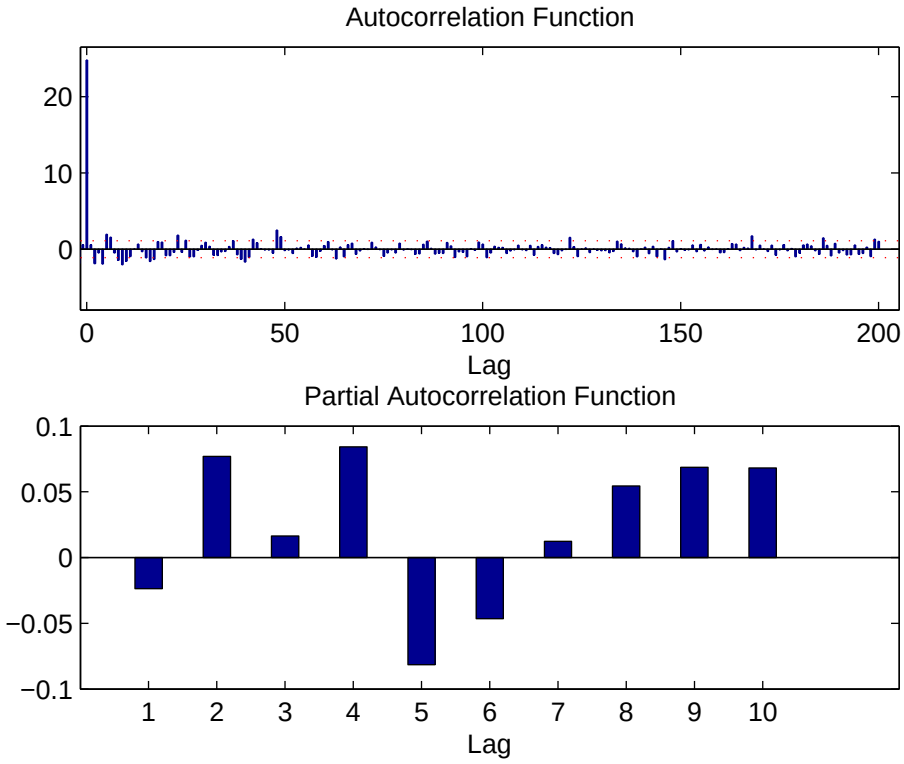


Figure 5.8: ACF and PACF Residual for Final Model

Finally, the fitted seasonal ARIMA model in Eq. (5.5) will be used to forecast the electricity load 1 hour ahead. The forecasts will be made on the training data. The results can be seen in the following figure.

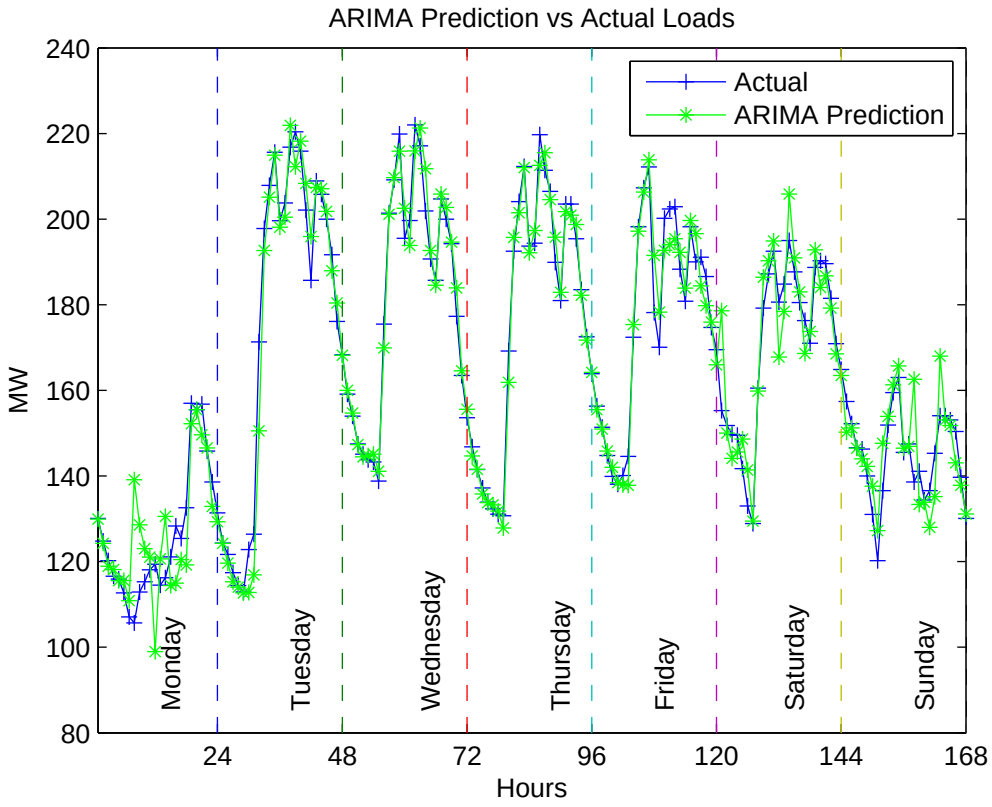


Figure 5.9: ARIMA Prediction for One Week in August - October, 2009

Fig. 5.9 shows a graph of the forecasts and the actual loads for a typical week. We see from this graph that there are poor forecasts at some points (e.g., at hours 8 and 12). There could be many causes for these large errors (e.g., a sudden shut down of nearby industrial plants, weekends, etc.). It is also possible that the performance could be improved by using a nonlinear model.

ARIMA models for the other training sets were also developed. The results are summarized in Table 5.4. We can notice from this table that the models are very similar throughout the year. This suggests that using one model for the entire year might be sufficient. We will consider this possibility again when we compare the accuracies of the various models (full year, six month, three month).

Table 5.4: ARIMA Models

Season	Time period	Model
Full year	May, 09 - April, 10	$(1+0.922B^{24})(1+0.885B^{168})\nabla_1\nabla_{24}z_t = (1+0.066B^{24})(1+0.992B^{168})e_t$
Dry Season	May, 09 - Oct, 09	$(1+0.914B^{24})(1+0.846B^{168})\nabla_1\nabla_{24}z_t = (1+0.069B^{24})(1+0.987B^{168})e_t$
Wet Season	Nov, 09 - April, 10	$(1+0.899B^{24})(1+0.893B^{168})\nabla_1\nabla_{24}z_t = (1+0.073B^{24})(1+1.008B^{168})e_t$
Dry Season 1	May, 09 - July, 09	$(1+0.937B^{24})(1+0.757B^{168})\nabla_1\nabla_{24}z_t = (1+0.030B^{24})(1+0.960B^{168})e_t$
Dry Season 2	August, 09 - Oct, 09	$(1+0.881B^{24})(1+0.787B^{168})\nabla_1\nabla_{24}z_t = (1+0.140B^{24})(1+0.970B^{168})e_t$
Wet Season 1	Nov, 09 - Jan, 10	$(1+0.867B^{24})(1+0.888B^{168})\nabla_1\nabla_{24}z_t = (1+0.088B^{24})(1+1.025B^{168})e_t$
Wet Season 2	Feb, 10 - April, 10	$(1+0.949B^{24})(1+0.883B^{168})\nabla_1\nabla_{24}z_t = (1+0.063B^{24})(1+1.033B^{168})e_t$

Table 5.5 shows the accuracy for the seasonal ARIMA models. The results show that the single (full year) model provides better forecasts than models developed for individual seasons. The two six-month models (dry and wet) produce a slightly better forecast during the wet season, but the accuracies are not significantly different.

Table 5.5: Seasonal ARIMA Performance for Training data

Season	Training Period	RMSE	MAE
Full year	May, 09 - April, 10	4.481	3.027
Dry	May, 09 - Oct, 09	4.677	3.199
Wet	Nov, 09 - April, 10	4.407	2.990
Dry 1	May, 09 - July, 09	4.777	3.250
Dry 2	August, 09 - Oct, 09	4.982	3.403
Wet 1	Nov, 09 - Jan, 10	4.513	3.115
Wet 2	Feb, 10 - April, 10	4.950	3.314

It is clear from these results that ARIMA models can provide reasonable forecasts of electricity consumption. However, ARIMA models have the drawback that they can only produce linear forecasts. In the next section, we want to try to increase the accuracy of short term forecasting by using a nonlinear model - the neural network.

5.3 Fitting the neural network model

As stated in Chapter 4, a good nonlinear model should be "general enough to capture some of the nonlinear phenomena in the data". One such model is the artificial neural network. In Chapter 4 we introduced a new neural network architecture - the PNARIMA network. In this section we describe how this network can be trained for load forecasting.

5.3.1 Preliminary structure determination

To build a neural network model for forecasting, we should pre-process the data first. The input to this neural network is the previous hourly electricity consumption. From the ACF and PACF behaviour shown in Fig. 5.4, we know that this is a non-stationary process, so we need to difference the data. Wang and Leu [49] suggested that neural networks trained with

differenced data can produce better predictions than those trained with raw data. Hence, we prefer to difference the data before training. The differenced load, w_t, is computed as in (5.6):

$$\begin{aligned} w_t &= \nabla_1^d \nabla_{24}^D z_t \\ w_t &= \nabla_1 \nabla_{24} z_t \\ w_t &= \nabla_{24} z_t - \nabla_{24} z_{t-1} \\ w_t &= z_t - z_{t-24} - z_{t-1} + z_{t-25} \end{aligned} \tag{5.6}$$

After being differenced, the data will be normalized to the interval -1 to 1.

Having pre-processed the data, we are ready to determine the network type. For our problem, we will use the PNARIMA model shown in Fig. 4.10. We will use the ARIMA model of (5.4) to determine the structure of the tapped delays in the PNARIMA network.

$$(1 - \Phi_1 B^{24} - \Phi_1 ' B^{168} + \Phi_1 \Phi_1 ' B^{192}) w_t = (1 - \Theta_1 B^{24} - \Theta_1 ' B^{168} + \Theta_1 \Theta_1 ' B^{192}) e_t$$

$$w_t - \Phi_1 B^{24} w_t - \Phi_1 ' B^{168} w_t + \Phi_1 \Phi_1 ' B^{192} w_t = e_t - \Theta_1 B^{24} e_t - \Theta_1 ' B^{168} e_t + \Theta_1 \Theta_1 ' B^{192} e_t$$

$$w_t - \Phi_1 w_{t-24} - \Phi_1 ' w_{t-168} + \Phi_1 \Phi_1 ' w_{t-192} = e_t - \Theta_1 e_{t-24} - \Theta_1 ' e_{t-168} + \Theta_1 \Theta_1 ' e_{t-192}$$

Using this ARIMA model, the architecture of the PNARIMA neural network is shown in Fig. 4.10. A more detailed diagram, showing the structure of the multilayer section, is shown in Fig. 5.10. The periodic tapped delay lines of Fig. 4.10 are collapsed to single tapped delays in Fig. 5.10.

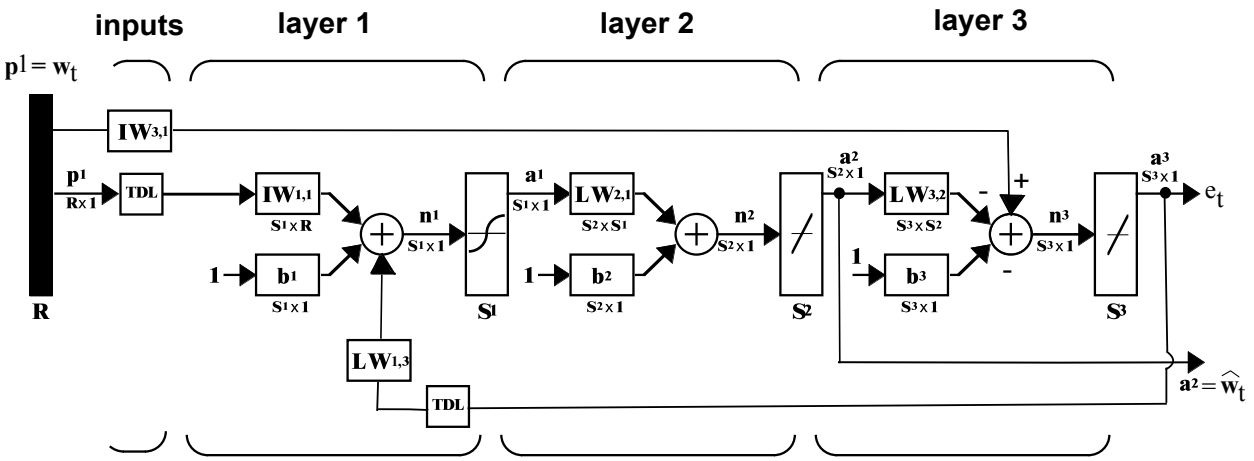


Figure 5.10: PNARIMA Architecture for Prediction

This network consists of three layers. The first two layers represent the multilayer network from Fig. 4.10. The third layer represents the summation in Fig. 4.10 that computes the prediction error. The weight $IW^{3,1} = 1$, the weight $LW^{3,2} = -1$ and the bias $b^3 = 0$. These parameters are fixed during training of the network. The delay for the input is based on the ARIMA model (5.4) so we have $DI_{1,1} = 24, 168, 192$ and $DL_{1,3} = 24, 168, 192$. The transfer function for the first layer is the hyperbolic tangent:

$$f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{5.7}$$

Linear transfer functions will be used in the second and third layers.

In initializing the weight and bias, we use the Nguyen-Widrow algorithm [45]. This neural network consists of two outputs. The first output is the error, e_t, which is the difference between actual value, w_t, and predicted value, $\hat{w}_t$, and the second output from layer two is the predicted value $\hat{w}_t$.

For the training algorithm, we use the Gauss-Newton approximation to Bayesian regularization [47]. Training is stopped when the norm of the gradient of mean squared error falls below a specified level, or when a maximum number of epochs is reached (100 for our tests).

5.3.2 Form of neural network prediction

We will fit the neural network to the differenced load

$$\begin{aligned} w_t &= \nabla_1 \nabla_{24} z_t = (1-B)(1-B^{24}) z_t = (1 - B - B^{24} + B^{25}) z_t \\ &= z_t - z_{t-1} - z_{t-24} + z_{t-25} \end{aligned} \tag{5.8}$$

Therefore, the trained network will produce predictions of the differenced load, $\hat{w}_t$. However, we need to convert this to a prediction of load, $\hat{z}_t$. From Eq. (5.8), we would expect that the relationship between $\hat{z}_t$ and $\hat{w}_t$ should be

$$\hat{z}_t = \hat{w}_t + z_{t-1} + z_{t-24} - z_{t-25} \tag{5.9}$$

In this section we will demonstrate that this equation does work for the linear ARIMA model. We will then use Eq. (5.9) for the PNARIMA predictor.

We can expand the ARIMA model in Eq.(5.4) to obtain

$$\begin{gathered} (1 - \Phi_1 B^{24} - \Phi_1 ` B^{168} + \Phi_1 \Phi_1 ` B^{192}) w_t = (1 - \Theta_1 B^{24} - \Theta_1 ` B^{168} + \Theta_1 \Theta_1 ` B^{192}) e_t \\ w_t - \Phi_1 B^{24} w_t - \Phi_1 ` B^{168} w_t + \Phi_1 \Phi_1 ` B^{192} w_t = e_t - \Theta_1 B^{24} e_t - \Theta_1 ` B^{168} e_t + \Theta_1 \Theta_1 ` B^{192} e_t \\ w_t - \Phi_1 w_{t-24} - \Phi_1 ` w_{t-168} + \Phi_1 \Phi_1 ` w_{t-192} = e_t - \Theta_1 e_{t-24} - \Theta_1 ` e_{t-168} + \Theta_1 \Theta_1 ` e_{t-192} \end{gathered} \tag{5.10}$$

We can solve for w_t as follows

$$w_t = \Phi_1 w_{t-24} + \Phi_1 ` w_{t-168} - \Phi_1 \Phi_1 ` w_{t-192} - (\Theta_1 e_{t-24} + \Theta_1 ` e_{t-168} - \Theta_1 \Theta_1 ` e_{t-192}) + e_t$$

The error, e_t, is the difference between the actual value and the prediction. It can be written as

$$e_t = w_t - \hat{w}_t$$

Hence

$$w_t = \Phi_1 w_{t-24} + \Phi_1{}' w_{t-168} - \Phi_1\Phi_1{}' w_{t-192} - \Theta_1(w_{t-24} - \hat{w}_{t-24}) \\ -\Theta_1{}'(w_{t-168} - \hat{w}_{t-168}) + \Theta_1\Theta_1{}'(w_{t-192} - \hat{w}_{t-192}) + e_t$$

This can also be written as

$$w_t = \hat{w}_t + e_t \tag{5.11}$$

where the value of $\hat{w}_t$ is

$$\begin{aligned} \hat{w}_t = {} & \Phi_1 w_{t-24} + \Phi_1{}' w_{t-168} - \Phi_1\Phi_1{}' w_{t-192} - \Theta_1(w_{t-24} - \hat{w}_{t-24}) \\ & - \Theta_1{}'(w_{t-168} - \hat{w}_{t-168}) + \Theta_1\Theta_1{}'(w_{t-192} - \hat{w}_{t-192}) \end{aligned} \tag{5.12}$$

Consider again our proposed $\hat{z}_t$ from Eq. (5.9): $\hat{z}_t = z_{t-1} + z_{t-24} - z_{t-25} + \hat{w}_t$. If we substitute $\hat{w}_t$ from (5.12) into this equation, we will get

$$\begin{aligned} \hat{z}_t &= z_{t-1} + z_{t-24} - z_{t-25} + \hat{w}_t \\ \hat{z}_t &= z_{t-1} + z_{t-24} - z_{t-25} + \Phi_1 w_{t-24} + \Phi_1{}' w_{t-168} - \Phi_1\Phi_1{}' w_{t-192} \\ &\quad - \Theta_1(w_{t-24} - \hat{w}_{t-24}) - \Theta_1{}'(w_{t-168} - \hat{w}_{t-168}) + \Theta_1\Theta_1{}'(w_{t-192} - \hat{w}_{t-192}) \end{aligned} \tag{5.13}$$

Since we know $w_t = z_t - z_{t-24} - z_{t-1} + z_{t-25}$ from (5.8), we can write the prediction of z_t in the above equation as

$$\begin{aligned} \hat{z}_t = {} & z_{t-1} + z_{t-24} - z_{t-25} + \Phi_1(z_{t-24} - z_{t-25} - z_{t-48} + z_{t-49}) + \Phi_1{}'(z_{t-168} - z_{t-169} - z_{t-192} \\ & + z_{t-193}) - \Phi_1\Phi_1{}'(z_{t-192} - z_{t-193} - z_{t-216} + z_{t-217}) - \Theta_1(w_{t-24} - \hat{w}_{t-24}) - \Theta_1{}'(w_{t-168} \\ & - \hat{w}_{t-168}) + \Theta_1\Theta_1{}'(w_{t-192} - \hat{w}_{t-192}) \end{aligned} \tag{5.14}$$

We want to write this expression in terms involving only z_t and $\hat{z}_t$. We can rewrite Eq. (5.9) as

$$\hat{w}_t = \hat{z}_t - z_{t-1} - z_{t-24} + z_{t-25}$$

So (5.14) can be rewritten as

$$\begin{aligned}
\hat{z}_t &= z_{t-1} + z_{t-24} - z_{t-25} + \Phi_1(z_{t-24} - z_{t-25} - z_{t-48} + z_{t-49}) \\
&+ \Phi_1`(z_{t-168} - z_{t-169} - z_{t-192} + z_{t-193}) - \Phi_1\Phi_1`(z_{t-192} - z_{t-193} - z_{t-216} + z_{t-217}) \\
&- \Theta_1(z_{t-24} - z_{t-25} - z_{t-48} + z_{t-49} - (\hat{z}_{t-24} - z_{t-25} - z_{t-48} + z_{t-49})) \\
&- \Theta_1`(z_{t-168} - z_{t-169} - z_{t-192} + z_{t-193} - (\hat{z}_{t-168} - z_{t-169} - z_{t-192} + z_{t-193}) \\
&+ \Theta_1\Theta_1`(z_{t-192} - z_{t-193} - z_{t-216} + z_{t-217} - (\hat{z}_{t-192} - z_{t-193} - z_{t-216} + z_{t-217})) \\
&= z_{t-1} + z_{t-24} - z_{t-25} + \Phi_1(z_{t-24} - z_{t-25} - z_{t-48} + z_{t-49}) + \Phi_1`(z_{t-168} - z_{t-169} - z_{t-192} \\
&+ z_{t-193}) - \Phi_1\Phi_1`(z_{t-192} - z_{t-193} - z_{t-216} + z_{t-217}) - \Theta_1(z_{t-24} - \hat{z}_{t-24}) \\
&- \Theta_1`(z_{t-168} - \hat{z}_{t-168}) + \Theta_1\Theta_1`(z_{t-192} - \hat{z}_{t-192})
\end{aligned} \tag{5.15}$$

This is the prediction of z_t for the ARIMA model, if we assume that (5.9) is the correct relationship between $\hat{z}_t$ and $\hat{w}_t$.

Now we will get the prediction of z_t by substituting the difference equation in (5.6) into the ARIMA model in (5.4) directly. If this produces the same form as (5.15), then (5.9) is the correct relationship between $\hat{z}_t$ and $\hat{w}_t$. The procedure is as follows

$$\begin{aligned}
(1 - \Phi_1 B^{24} - \Phi_1` B^{168} + \Phi_1\Phi_1` B^{192})\nabla_1\nabla_{24} z_t &= (1 - \Theta_1 B^{24} - \Theta_1` B^{168} \\
&+ \Theta_1\Theta_1` B^{192}) e_t \\
(1 - \Phi_1 B^{24} - \Phi_1` B^{168} + \Phi_1\Phi_1` B^{192})(1 - B^1 - B^{24} + B^{25}) z_t &= (1 - \Theta_1 B^{24} - \Theta_1` B^{168} \\
&+ \Theta_1\Theta_1` B^{192}) e_t \\
(1 - \Phi_1 B^{24} - \Phi_1` B^{168} + \Phi_1\Phi_1` B^{192} - B + \Phi_1 B^{25} + \Phi_1` B^{169} - \Phi_1\Phi_1` B^{193} - B^{24} + \Phi_1 B^{48} \\
+ \Phi_1` B^{192} - \Phi_1\Phi_1` B^{216} + B^{25} - \Phi_1 B^{49} - \Phi_1` B^{193} + \Phi_1\Phi_1` B^{217}) z_t &= (1 - \Theta_1 B^{24} \\
- \Theta_1` B^{168} + \Theta_1\Theta_1` B^{192}) e_t
\end{aligned}$$

If we expand the above equation, we can get

$$z_t - \Phi_1 B^{24} z_t - \Phi_1 ` B^{168} z_t + \Phi_1 \Phi_1 ` B^{192} z_t - B z_t + \Phi_1 B^{25} z_t + \Phi_1 ` B^{169} z_t -$$
$$\Phi_1 \Phi_1 ` B^{193} z_t - B^{24} z_t + \Phi_1 B^{48} z_t + \Phi_1 ` B^{192} z_t - \Phi_1 \Phi_1 ` B^{216} z_t + B^{25} z_t - \Phi_1 B^{49} z_t +$$
$$\Phi_1 ` B^{193} z_t + \Phi_1 \Phi_1 ` B^{217} z_t = e_t - \Theta_1 B^{24} e_t - \Theta_1 ` B^{168} e_t + \Theta_1 \Theta_1 ` B^{192} e_t$$
$$z_t - B z_t - B^{24} z_t + B^{25} z_t = \Phi_1 B^{24} z_t - \Phi_1 B^{25} z_t - \Phi_1 B^{48} z_t + \Phi_1 B^{49} z_t + \Phi_1 ` B^{168} z_t -$$
$$\Phi_1 ` B^{169} z_t - \Phi_1 ` B^{192} z_t + \Phi_1 ` B^{193} z_t - \Phi_1 \Phi_1 ` B^{192} z_t + \Phi_1 \Phi_1 ` B^{193} z_t +$$
$$\Phi_1 \Phi_1 ` B^{216} z_t - \Phi_1 \Phi_1 ` B^{217} z_t - \Theta_1 B^{24} e_t - \Theta_1 ` B^{168} e_t +$$
$$\Theta_1 \Theta_1 ` B^{192} e_t + e_t$$
$$z_t - z_{t-1} - z_{t-24} + z_{t-25} = \Phi_1 (z_{t-24} - z_{t-25} - z_{t-48} + z_{t-49}) + \Phi_1 ` (z_{t-168} - z_{t-169} -$$
$$z_{t-192} + z_{t-193}) - \Phi_1 \Phi_1 ` (z_{t-192} - z_{t-193} - z_{t-216} + z_{t-217}) - (\Theta_1 e_{t-24} +$$
$$\Theta_1 ` e_{t-168} - \Theta_1 \Theta_1 ` e_{t-192}) + e_t$$
$$z_t = z_{t-1} + z_{t-24} - z_{t-25} + \Phi_1 (z_{t-24} - z_{t-25} - z_{t-48} + z_{t-49}) + \Phi_1 ` (z_{t-168} - z_{t-169} -$$
$$z_{t-192} + z_{t-193}) - \Phi_1 \Phi_1 ` (z_{t-192} - z_{t-193} - z_{t-216} + z_{t-217}) - (\Theta_1 e_{t-24} +$$
$$\Theta_1 ` e_{t-168} - \Theta_1 \Theta_1 ` e_{t-192}) + e_t$$

Therefore, the prediction of z_t is

$$\begin{aligned}
\hat{z_t} &= z_{t-1} + z_{t-24} - z_{t-25} + \Phi_1 (z_{t-24} - z_{t-25} - z_{t-48} + z_{t-49}) + \\
&\Phi_1 ` (z_{t-168} - z_{t-169} - z_{t-192} + z_{t-193}) - \Phi_1 \Phi_1 ` (z_{t-192} - z_{t-193} - z_{t-216} + z_{t-217}) - \\
&(\Theta_1 e_{t-24} + \Theta_1 ` e_{t-168} - \Theta_1 \Theta_1 ` e_{t-192}) \\
\hat{z_t} &= z_{t-1} + z_{t-24} - z_{t-25} + \Phi_1 (z_{t-24} - z_{t-25} - z_{t-48} + z_{t-49}) + \\
&\Phi_1 ` (z_{t-168} - z_{t-169} - z_{t-192} + z_{t-193}) - \Phi_1 \Phi_1 ` (z_{t-192} - z_{t-193} - z_{t-216} + z_{t-217}) \\
&- \Theta_1 (z_{t-24} - \hat{z}_{t-24}) - \Theta_1 ` (z_{t-168} - \hat{z}_{t-168}) + \Theta_1 \Theta_1 ` (z_{t-192} - \hat{z}_{t-192})
\end{aligned} \tag{5.16}$$

This result is equivalent with the previous equation (5.15). Hence, we can conclude that

(5.9) is correct:

$$\hat{z}_t = z_{t-1} + z_{t-24} - z_{t-25} + \hat{w}_t \tag{5.17}$$

This technique for calculating the prediction $\hat{z}_t$ from $\hat{w}_t$ in Eq. (5.17) can be adapted to the neural network model forecast.

For the neural network of Fig. 5.10, $\hat{w}_t$ is the output from the second layer. Hence, the nonlinear prediction can be written as

$$\hat{z}_t = z_{t-1} + z_{t-24} - z_{t-25} + \text{Neural Network Result from the second layer} \tag{5.18}$$

5.3.3 Fitting and validation of the neural network model

As with the ARIMA model, we will begin neural network training with the Dry 2 season (August to October, 2009). To check the adequacy of the model, we can investigate the autocorrelation function of the residuals (prediction errors) of the fitted model. The ACF of the residuals for the training data from August to October, 2009, is shown in Fig. 5.11.

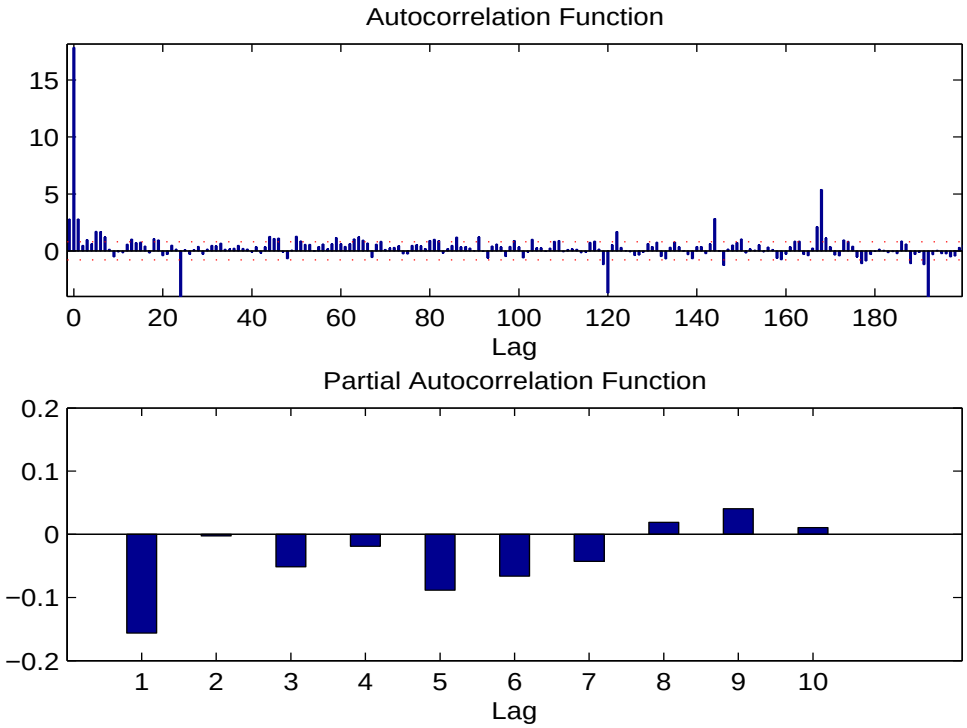


Figure 5.11: ACF and PACF Residual for PNARIMA Neural Network

The ACF shows that the forecast error is almost uncorrelated (white noise). This indicates that the model is adequate. Fig. 5.12 shows the actual and forecasted loads using the neural network model for the training data in a typical week between August and October, 2009. Table 5.6 shows the network performance for the training data.

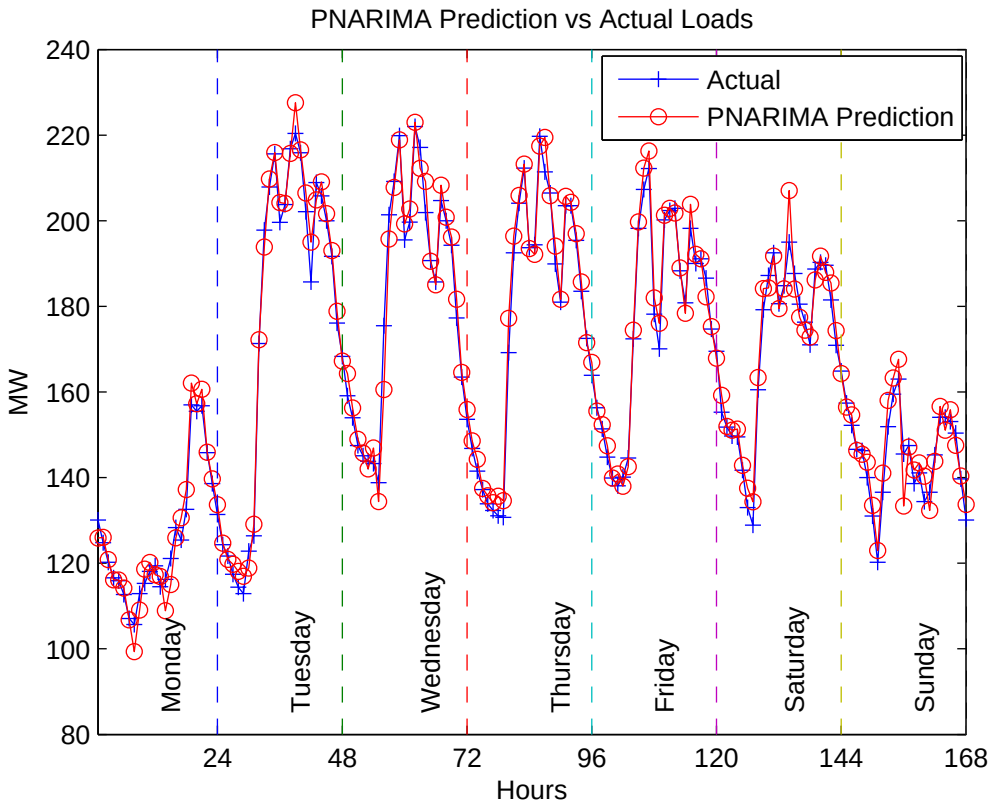


Figure 5.12: PNARIMA Prediction During a Typical Week During August - October, 2009

Table 5.6: PNARIMA Performance for Training data

Season	Training Period	RMSE	MAE
Full year	May, 09 - April, 10	4.427	3.043
Dry	May, 09 - Oct, 09	4.650	3.207
Wet	Nov, 09 - April, 10	4.215	2.934
Dry 1	May, 09 - July, 09	4.700	3.260
Dry 2	August, 09 - Oct, 09	4.227	3.025
Wet 1	Nov, 09 - Jan, 10	3.749	2.603
Wet 2	Feb, 10 - April, 10	4.219	2.838

From the network performance, we can see that the PNARIMA predictor is better than the ARIMA predictor (see Table 5.5). It means that there are nonlinear aspects of the load process that the neural network has the capability to capture, but that the linear ARIMA model does not.

5.4 Comparison of ARIMA and neural network models on test data

Having developed the linear ARIMA model and the PNARIMA neural network, we are ready to implement these models on the testing data. The objective is to evaluate the model performance. The testing data is the hourly electricity consumption over the period May, 2010 to April, 2011. There are two months (September, 2010 and January, 2011) which are missed from the testing data set.

Fig. 5.13 shows the actual load and ARIMA and PNARIMA predictions for a typical week during the testing period. The prediction models were fit using training data from August to October, 2009, and the predictions in Fig. 5.13 are from the period August to October, 2010.

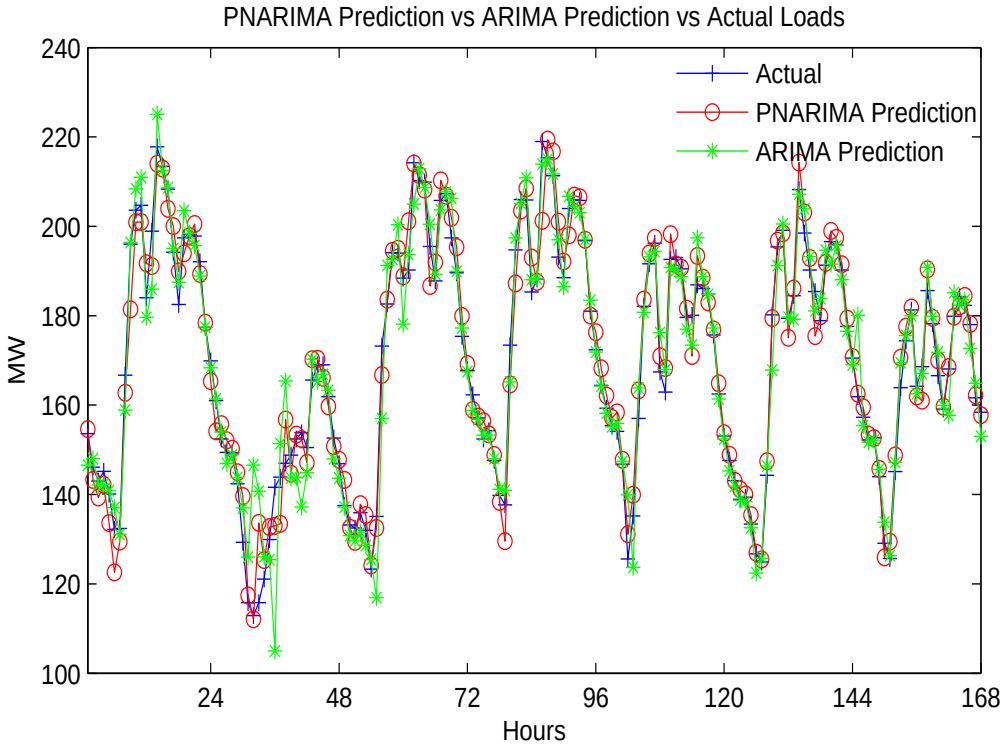


Figure 5.13: PNARIMA and ARIMA Predictions of Testing Data from The Period from August to October, 2010

In this testing period, the ARIMA error is worse than the PNARIMA error. The performance of the PNARIMA predictor is compared with the ARIMA performance for the other testing data in Table 5.7.

There are a couple of conclusions we can draw from the testing results. First, the PNARIMA predictor produces better results than the ARIMA predictor in all cases. This means that the load process is clearly nonlinear. In addition, it appears that the process does not change in a significant way throughout the year. The errors for the single model that was fit for the entire year are very similar to those for models that were fit over only six or three month periods. If we consider the total RMSE over the entire year, the single model has RMSE of 4.835. The separate wet and dry models have a total RMSE over the year of 4.844. The four subseason models (Dry 1, Dry 2, Wet 1, Wet 2) have a combined 4.631 RMSE over the test year.

Table 5.7: ARIMA and PNARIMA Performance for Testing Data

Forecasting Period	Method	RMSE	MAE
May, 10 - April, 11	PNARIMA	4.835	3.380
Full	ARIMA	5.214	3.614
May, 10 - Oct, 10	PNARIMA	4.938	3.411
Dry	ARIMA	5.245	3.637
Nov, 10 - April, 11	PNARIMA	4.752	3.376
Wet	ARIMA	5.261	3.629
May, 10 - July, 10	PNARIMA	4.847	3.432
Dry 1	ARIMA	5.028	3.498
August, 10 - Oct, 10	PNARIMA	4.645	3.389
Dry 2	ARIMA	5.672	3.928
Nov, 10 - Dec, 10	PNARIMA	4.699	3.249
Wet 1	ARIMA	5.612	3.887
Feb, 11 - April, 11	PNARIMA	4.333	3.025
Wet 2	ARIMA	5.181	3.630

CHAPTER 6

CONCLUSIONS

The objective of this research has been the development of accurate short term forecasts of electric power loads. We began by using linear ARIMA models. These models were able to capture the non-stationarity of the electric load process. The seasonal ARIMA model is relatively easy to develop, using some systematic procedures, and it produces good forecasts. This makes the linear approach popular for short term load forecasting.

Although the ARIMA model generally produces good results, it has a drawback. It assumes a linear relationship between present and future values of load and between weather variables and load. Based on this consideration, we need a nonlinear approach to achieve better performance. Since ARIMA models can capture non-stationary and linear factors, and since neural networks can capture nonlinear effects, we have decided to combine these two approaches to produce a periodic nonlinear ARIMA (PNARIMA) neural network. This is a new approach to short term load forecasting with neural networks, because it considers not only the autoregressive component, but also the moving average component (the addition of the moving average component requires that the network be trained with dynamic backpropagation, which is not needed for the purely autoregressive neural networks that have been used in the past). In addition, the PNARIMA network uses periodic tapped delay lines to account for the seasonality in the load process, and it also includes differencing, which allows the model to handle the non-stationarity in the load process.

We have tested our linear and nonlinear models using the actual electricity load consumption from PT. PLN Batam, Indonesia. We used two years of data from May 2009 to April 2011. (The first year was used for training the models. The second year was used for

testing.) The main input for this study is the previous electricity load consumption. We did not include the weather variables in our models because we found that they did not significantly improve the forecasts. The temperature does not strongly influence the electricity consumption, since this area has a tropical rain forest climate that has similar temperatures for wet and dry seasons. For the purpose of this study, we divided the historical load data into training and testing data. The training data is used to estimate the model parameters and the testing data is used to validate the model and to predict future model performance.

We fit several ARIMA and PNARIMA models : one for the whole year, one for each season (dry and wet) and one for each sub season (dry 1, dry 2, wet 1 and wet 2). ARIMA model development follows three steps: preliminary identification, parameter estimation and diagnostic testing. Preliminary identification determined that the best seasonal ARIMA model is $(0,1,0) \times (1,1,1)_{24} \times (1,0,1)_{168}$. This structure was also used for the PNARIMA model. The best neural network model had five neurons in the hidden layer, with tangent sigmoid transfer function in the hidden layer and linear transfer function for the output layers. The input to the neural network is the differenced load data. For the training data, neural network performance was better than ARIMA performance. This indicates that the PNARIMA model has the capability to capture the nonlinear aspects that cannot be captured by the ARIMA model.

The models were tested using data from the year following the training data. The results on the testing data showed that the performance of PNARIMA neural networks is better than linear ARIMA models for all cases. For example, PNARIMA RMSE is 4.645 and ARIMA RMSE is 5.672 for the dry 2 model testing data. For the full year model, PNARIMA RMSE is 4.835 and ARIMA RMSE is 5.214. This is an improvement of up to 5% in accuracy, which will be meaningful in terms of operating costs. These results indicate that the load process is clearly nonlinear. In addition, the process does not change significantly throughout the year. The errors for the single model that was fit for the entire year are very similar to those models that were fit over only six or three month periods.

For future work, the newly proposed PNARIMA model can be implemented on another utility system for a region where electric load consumption is influenced by weather variables, such as temperature, humidity, wind, rainfall, etc. The PNARIMA model can be easily adjusted, as described in Chapter 4, to include external inputs, like weather variables.

This new approach can also be applied to predict wind speed, which would be useful for wind power prediction.

REFERENCES

[1] D. W. Bunn and E. D. Farmer, *Comparative models for electrical load forecasting*. John Wiley and Sons Inc., 1985.

[2] P. Mandal, T. Senjyu, N. Urasaki, and T. Funabashi, "A neural network based several-hour-ahead electric load forecasting using similar days approach," *International Journal Electrical Power and Energy Systems*, vol. 28, pp. 367–373, July 2006.

[3] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks or short-term load forecasting: A review and evaluation," *IEEE Trans. on Power Systems*, vol. 16, pp. 44–55, February 2001.

[4] M. Hagan and R. Klein, "Identification techiques of Box and Jenkins applied to the problem of short term load forecasting," *1977 Summer Meeting of the IEEE Power Engineering Society*, pp. 618–612, July 1977.

[5] A. D. Papalexopoulos and T. C. Hesterberg, "A regression based approach to short term system load forecasting," *IEEE Trans. on Power Systems*, vol. 5, pp. 1535–1550, November 1990.

[6] K. Kalaitzakis, G. S. Stavrakakis, and E. M. Anagnostakis, "Short-term load forecasting based on artificial neural networks parallel implementation," *Electric Power Systems Research*, vol. 63, pp. 185–196, October 2002.

[7] M. Hagan and R. Klein, "On-line maximum likelihood estimation for load forecasting," *IEEE Trans. on System, Man and Cybernetics*, vol. SMC-8, pp. 711–715, September 1978.

[8] M. T. Hagan and S. M. Behr, "The time series approcah to short term load forecasting," *IEEE Trans. on Power Systems*, vol. 2, pp. 785–791, August 1987.

[9] Y. Zhang, *Linear and nonlinear modeling and forecasting of electric power loads*. PhD thesis, Oklahoma State University, 1992.

[10] K. L. Ho, Y. Y. Hsu, C. F. Chen, T. E. Lee, C. C. Liang, T. S. Lai, and K. K. Chen, "Short term load forecasting of Taiwan power system using a knowledge based expert system," *IEEE Trans. on Power Systems*, vol. 5, pp. 1214–1221, November 1990.

[11] M. Adya and F. Collopy, "How efective are neural networks at forecasting and prediction? a review and evaluation," *Journal of forecasting*, vol. 17, no. 4, pp. 481–495, 1998.

[12] G. Zhang, E. B. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: the state of the art," *International Journal of Forecasting*, vol. 14, pp. 35–62, March 1998.

[13] T. M. Peng, N. F. Hubele, and G. G. Karady, "Advancement in the application of neural networks for short term load forecasting," *IEEE Trans. on Power Systems*, vol. 7, pp. 250–257, February 1992.

[14] T. Senjyu, H. Takara, K. Uezato, and T. Funabashi, "One-hour-ahead forecasting using neural network," *IEEE Trans. on Power Systems*, vol. 17, pp. 113–118, February 2002.

[15] J. N. Fidalgo and M. A. Matos, "Forecasting portugal global load with artificial neural networks," *Proceeding ICANN'07 Proceedings of the 17th international conference on Artificial neural networks*, pp. 728–737, 2007.

[16] G. Gross and F. D. Galiana, "Short-term load forecasting," *Proceedings of the IEEE*, vol. 75, pp. 1558–1573, December 1987.

[17] E. Kyriakides and M. Polycarpou, *Short Term Electric Load Forecasting: A Tutorial*, vol. 35 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 2007.

[18] H. Hahn, S. MeyerNieberg, and S. Pickl, "Electric load forecasting methods: Tools for decision making," *European Journal of Operational Research*, vol. 199, pp. 902–907, December 2009.

[19] G. J. Tsekouras, N. D. Hatziargyriou, and E. N. Dialynas, "An optimized adaptive neural network for annual midterm energy forecasting," *IEEE Trans. on Power Systems*, vol. 21, pp. 385–391, February 2006.

[20] E. Doveh, P. Feigin, D. Greig, and L. Hyams, "Experience with FNN models for medium term power demand predictions," *IEEE Trans. on Power Systems*, vol. 14, pp. 538–546, May 1999.

[21] J. Reneses, E. Centeno, and J. Barquin, "Coordination between medium-term generation planning and short-term operation in electricity markets," *IEEE Trans. on Power Systems*, vol. 21, pp. 43–52, February 2006.

[22] M. S. Kandil, S. M. El-Debeiky, and N. E. Hasanien, "Long-term load forecasting for fast developing utility using a knowledge-based expert system," *IEEE Trans. on Power Systems*, vol. 5, pp. 1214–1221, November 2002.

[23] E. A. Feinberg and D. Genethliou, *Chapter 12: Load forecasting*. Power Electronics and Power Systems, Springer US, 2005.

[24] J. Y. Fan and J. D. McDonald, "A real-time implementation of short term load forecasting for distribution power systems," *IEEE Trans. on Power Systems*, vol. 9, pp. 988–994, May 1994.

[25] M. Espinoza, C. Joye, R. Belmans, and B. D. Moor, "Short-term load forecasting, profile identification,and customer segmentation: A methodology based on periodic time series," *IEEE Trans. on Power Systems*, vol. 20, pp. 1622–1630, August 2005.

[26] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, January 2003.

[27] G. A. N. Mbamalu and M. E. El-Hawary, "Load forecasting via suboptimal seasonal autoregressive models and iteratively reweighted least squares estimation," *IEEE Trans. on Power Systems*, vol. 8, pp. 343–348, February 1993.

[28] T. Haida and S. Muto, "Regression based peak load forecasting using a transformation technique," *IEEE Trans. on Power Systems*, vol. 9, pp. 1788–1794, November 1994.

[29] J. H. Park, Y. M. Park, and K. Y. Lee, "Composite modeling for adaptive short term load forecasting," *IEEE Trans. on Power Systems*, vol. 6, pp. 450–457, May 1991.

[30] H. M. Al-Hamadi and S. A. Soliman, "Short-term electric load forecasting based on kalman filtering algorithm with moving window weather and load model," *Electric Power Systems Reasearch*, 2004.

[31] S. Sargunaraj, D. P. S. Gupta, and S. Devi, "Short-term load forecasting for demand side management," *IEE Proceedings on Generation, Transmission and Distribution*, vol. 144, pp. 68–74, January 1997.

[32] B. F. Hobbs, S. Jitprapaikulsarn, and D. J. Maratukulam, "Analysis of the value for unit commitment of improved load forecasts," *IEEE Trans. on Power Systems*, vol. 14, pp. 1342–1348, November 1999.

[33] F. Zhao and H. Su, "Short-term load forecasting using kalman filter and elman neural network," *IEEE Conference on Industrial Electronics and Applications*, pp. 1043–1047, May 2007.

[34] S. Rahman and O. Hazim, "A generalized knowledge-based short-term load-forecasting technique," *IEEE Trans. on Power Systems*, vol. 8, pp. 508–514, May 1993.

[35] S. J. Kiartzis and A. G. Bakirtzis, "A fuzzy expert system for peak load forecasting: application to the greek power system," *Proceedings of the 10th Mediterranean Electrotechnical Conference*, vol. 3, pp. 1097–1100, May 2000.

[36] D. Srinivasan, S. S. Tan, C. S. Chang, and E. K. Chan, "Parallel neural network-fuzzy expert system strategy for short term load forecasting: system implementation and performance evaluation," *IEEE Trans. on Power Systems*, vol. 14, pp. 1100–1106, August 1999.

[37] B. J. Chen, M. W. Chang, and C. J. Lin, "Load forecasting using support vector machines: A study on EUNITE competition 2001," *IEEE Trans. on Power Systems*, vol. 19, pp. 1821–1830, November 2004.

[38] D. C. Montgomery, C. L. Jennings, and M. Kulahci, *Introduction to time series analysis and forecasting*. John Wiley & Sons, Inc., 2008.

[39] G. P. Box, G. M. Jenkins, and G. Reisnsel, *Time Series Analysis: Forecasting and Control*. Prentice Hall, Englewood Cliffs, 3rd ed., 1994.

[40] K. H. Chan, J. C. Hayya, and J. K. Ord, "A note on trend removal methods: the case of polynomial regression versus variate differencing," *Econometrica*, vol. 45, pp. 737–744, April 1977.

[41] W. A. Woodward, H. L. Gray, and A. C. Elliot, *Applied time series*. CRC Press Taylor & Francis Group, 2012.

[42] J. G. D. Gooijer, B. M. Wiliamowski, and L. . Weatherford, "Some recent development in non linear time series modelling, testing and forecasting," *International Journal of Forecasting*, vol. 8, pp. 135–156, October 1992.

[43] M. Hagan, H. Demuth, and M. Beale, *Neural networks design*. PWS Publishing Co, 3rd ed., 1996.

[44] S. Chen, S. A. Billings, and P. M. Grant, "Non linear system identification using neural networks," *International Journal of Control*, vol. 51, pp. 1191–1214, August 1990.

[45] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," *Proceeding of IJCNN*, vol. 3, pp. 21–26, June 1990.

[46] W. S. Sarle, "Stopped training and other remedies for overfitting," *Proceeding of the 27th Symposium on the Interface*, 1995.

[47] F. D. Foresee and M. Hagan, "Gauss-Newton approximation to bayesian learning," *Proceedings of the 1997 International Joint Conference on Neural Networks*, vol. 3, pp. 1930–1935, June 1997.

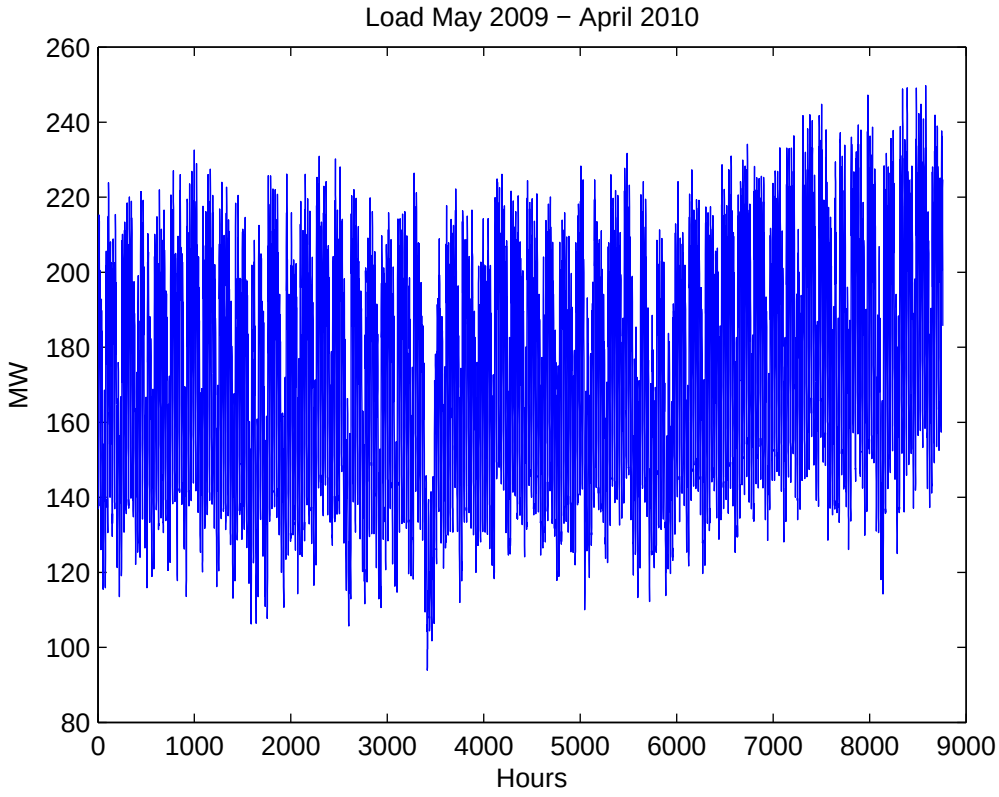
[48] PT. PLN Batam, "Statistik perusahaan." `http://info.plnbatam.com/info/index1.php?page=statistik-perusahaan`. 2012 (accessed March 19, 2013).

[49] J. H. Wang and J. Y. Leu, "Stock market trend prediction using ARIMA-based neural networks," *IEEE Int. Conf. on Neural Networks*, vol. 4, pp. 2160–2165, June 1996.

APPENDIX (Training and Testing Result)

Training data

Training data 1 (Full year - May 2009 to April 2010)



Load May 2009 to April 2010

The ARIMA model for the training data set May 2009 to April 2010

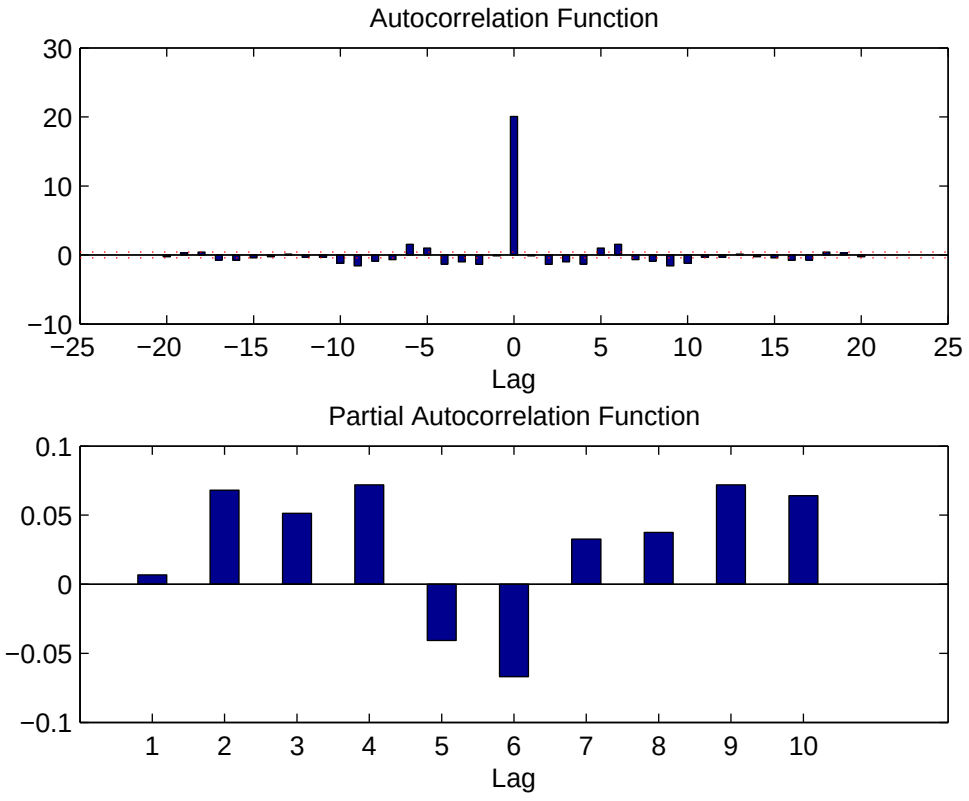
$$(1 + 0.922B^{24})(1 + 0.885B^{168})\nabla^1\nabla^{24}z_t = (1 + 0.066B^{24})(1 + 0.992B^{168})e_t$$

The confidence interval for the training data set May 2009 to April 2010

Confidence Interval for Final ARIMA Model

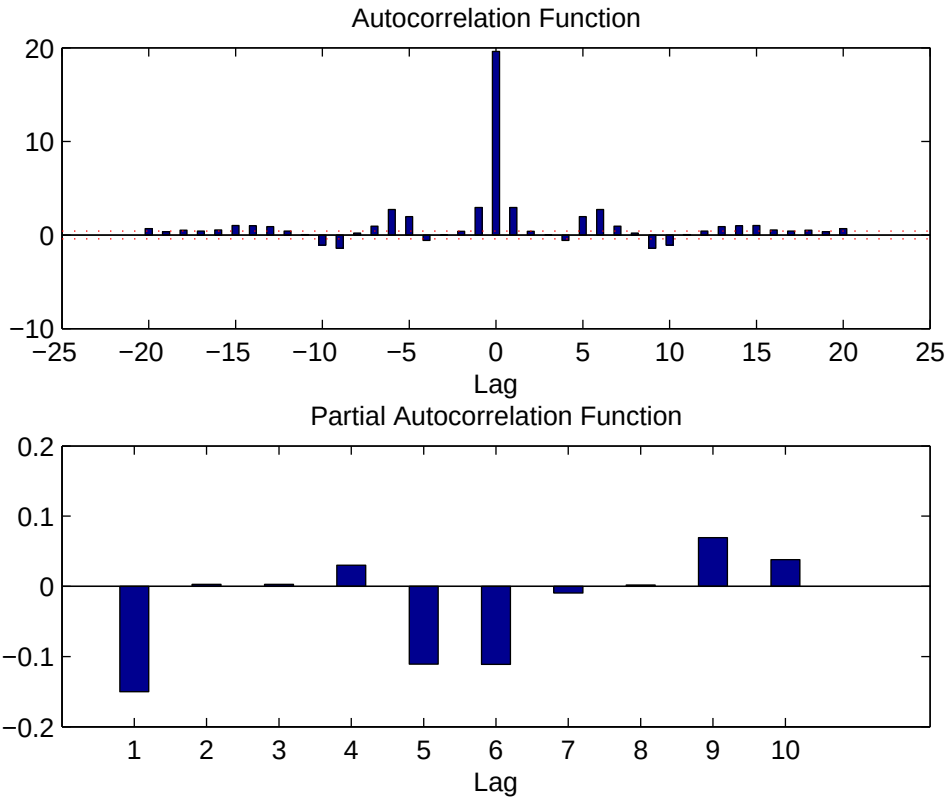
Parameters	Confidence Interval	
	Lower Bound	Upper Bound
Φ_1	-0.913	-0.931
Φ'_1	-0.870	-0.900
Θ_1	-0.043	-0.090
Θ_1‘	-0.985	-0.998

The ACF and PACF for the residual of the ARIMA final model training data set May 2009 to April 2010



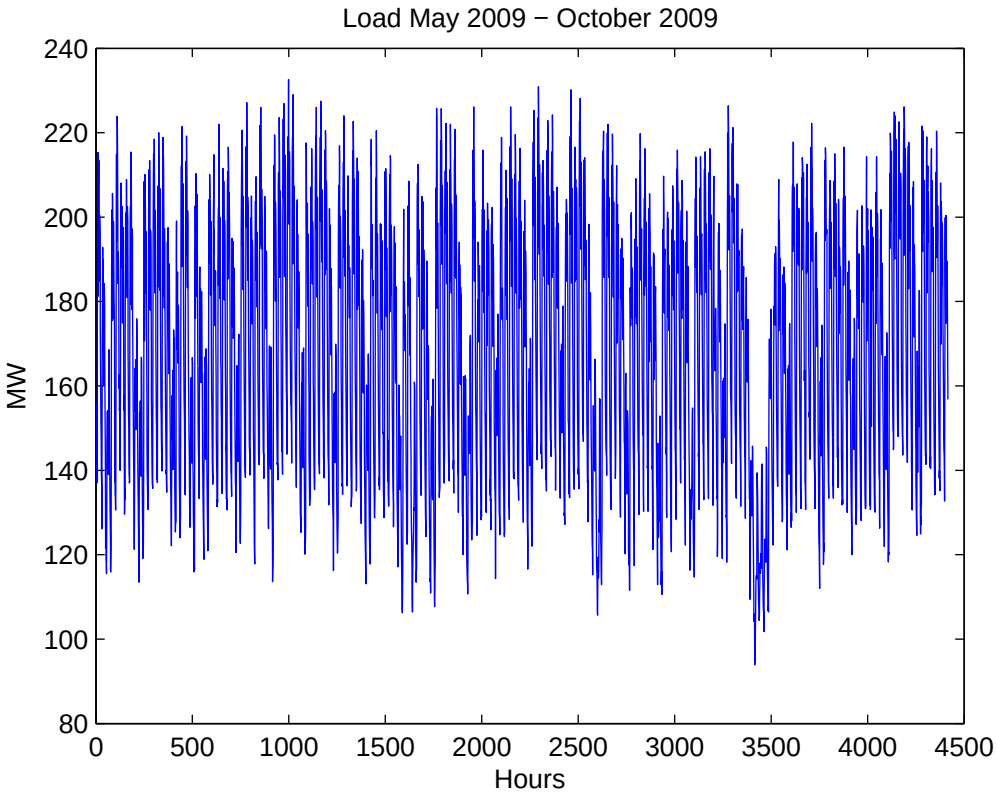
ACF and PACF Residual for Final Model

The ACF and PACF for the residual of PNARIMA neural network for training data set May 2009 to April 2010



ACF and PACF Residual for Final Model

Training data 2 (Dry - May 2009 to October 2009)



Load May 2009 to October 2009

The ARIMA model for the training data set May 2009 to October 2009

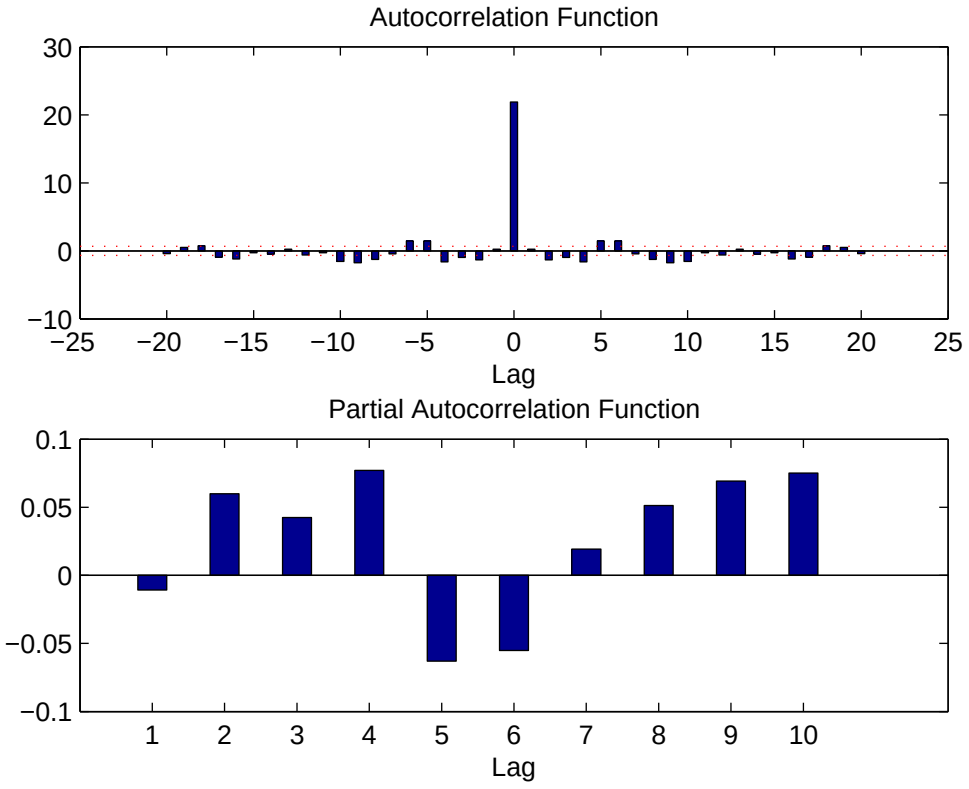
$$(1 + 0.914B^{24})(1 + 0.846B^{168})\nabla^{1}\nabla^{24}z_t = (1 + 0.069B^{24})(1 + 0.987B^{168})e_t$$

The confidence interval for the training data set May 2009 to October 2009

Confidence Interval for Final ARIMA Model

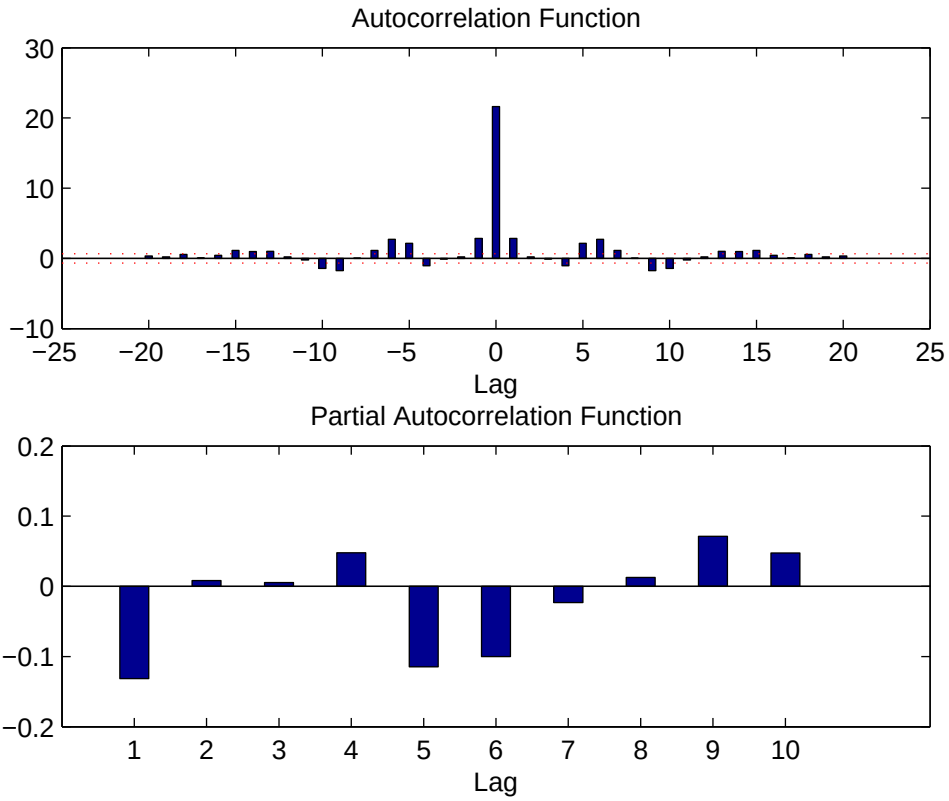
Parameters	Confidence Interval	
	Lower Bound	Upper Bound
Φ_1	-0.900	-0.928
Φ'_1	-0.817	-0.874
Θ_1	-0.036	-0.103
Θ_1‘	-0.973	-1.001

The ACF and PACF for the residual of the ARIMA final model training data set May 2009 to October 2009



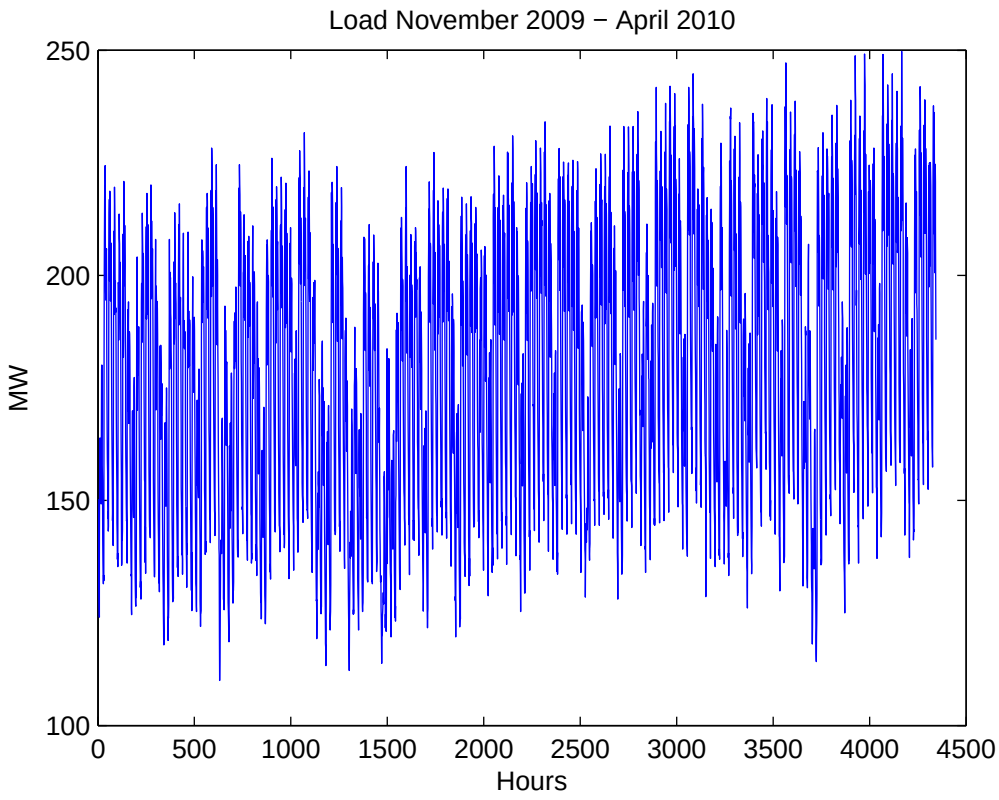
ACF and PACF Residual for Final Model

The ACF and PACF for the residual of PNARIMA neural network for training data set May 2009 to October 2009



ACF and PACF Residual for Final Model

Training data 3 (Wet - November 2009 to April 2010)



Load November 2009 to April 2010

The ARIMA model for the training data set November 2009 to April 2010

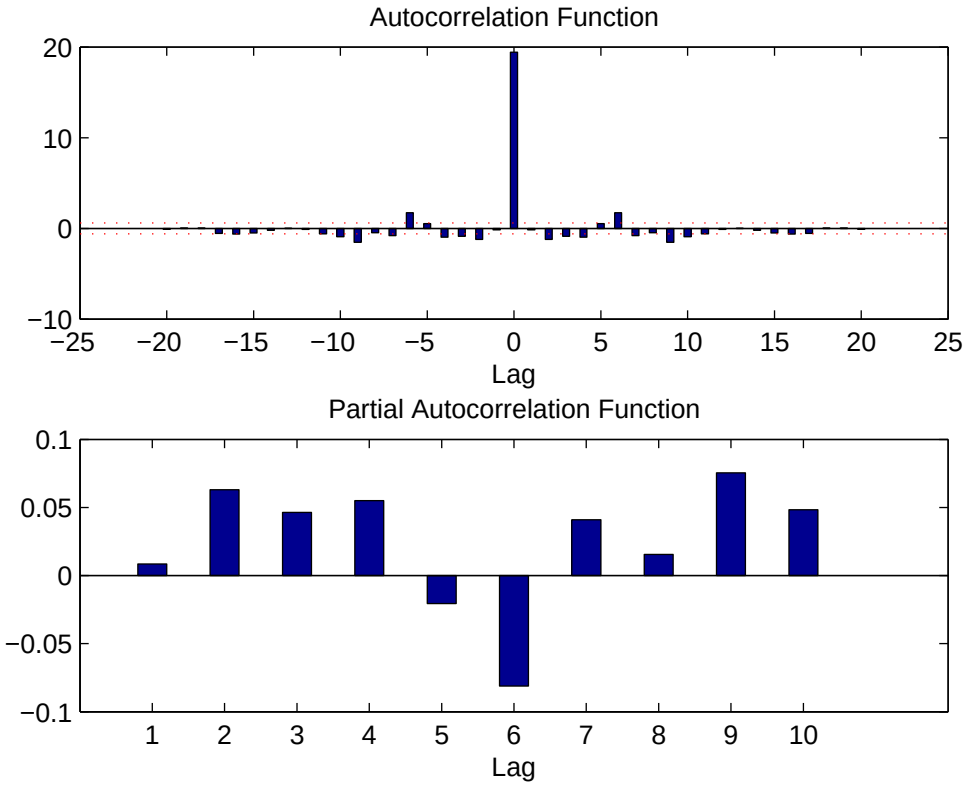
$$(1 + 0.899B^{24})(1 + 0.893B^{168})\nabla^1\nabla^{24}z_t = (1 + 0.073B^{24})(1 + 1.008B^{168})e_t$$

The confidence interval for the training data set November 2009 to April 2010

Confidence Interval for Final ARIMA Model

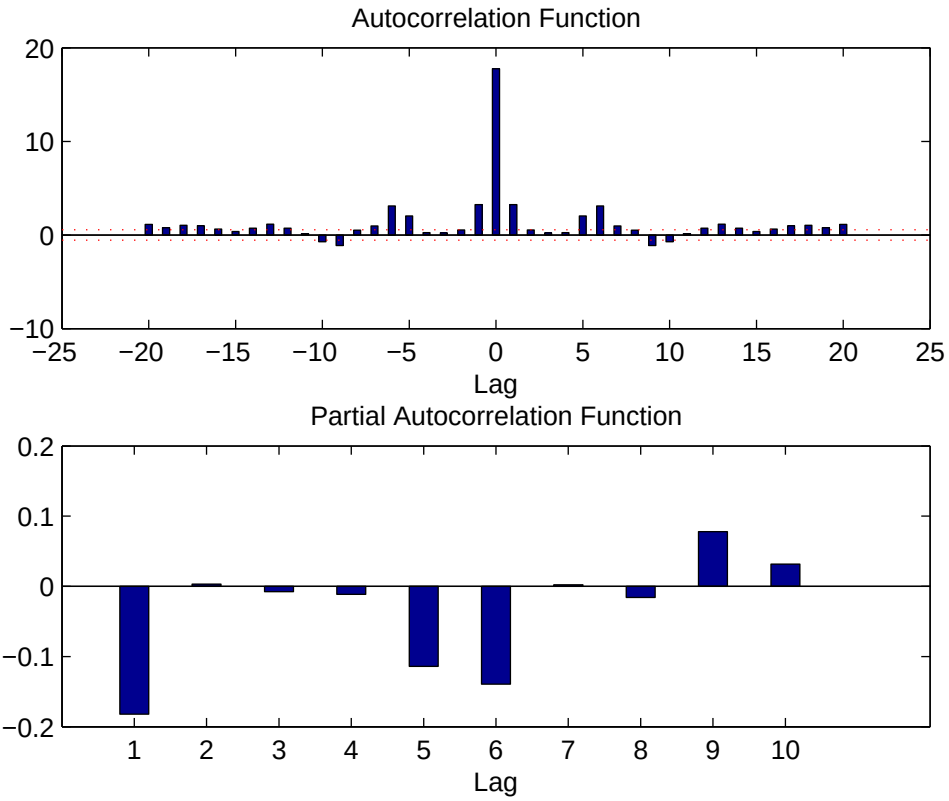
Parameters	Confidence Interval	
	Lower Bound	Upper Bound
Φ_1	-0.883	-0.914
Φ'_1	-0.866	-0.919
Θ_1	-0.038	-0.107
Θ_1‘	-0.996	-1.021

The ACF and PACF for the residual of the ARIMA final model training data set November 2009 to April 2010



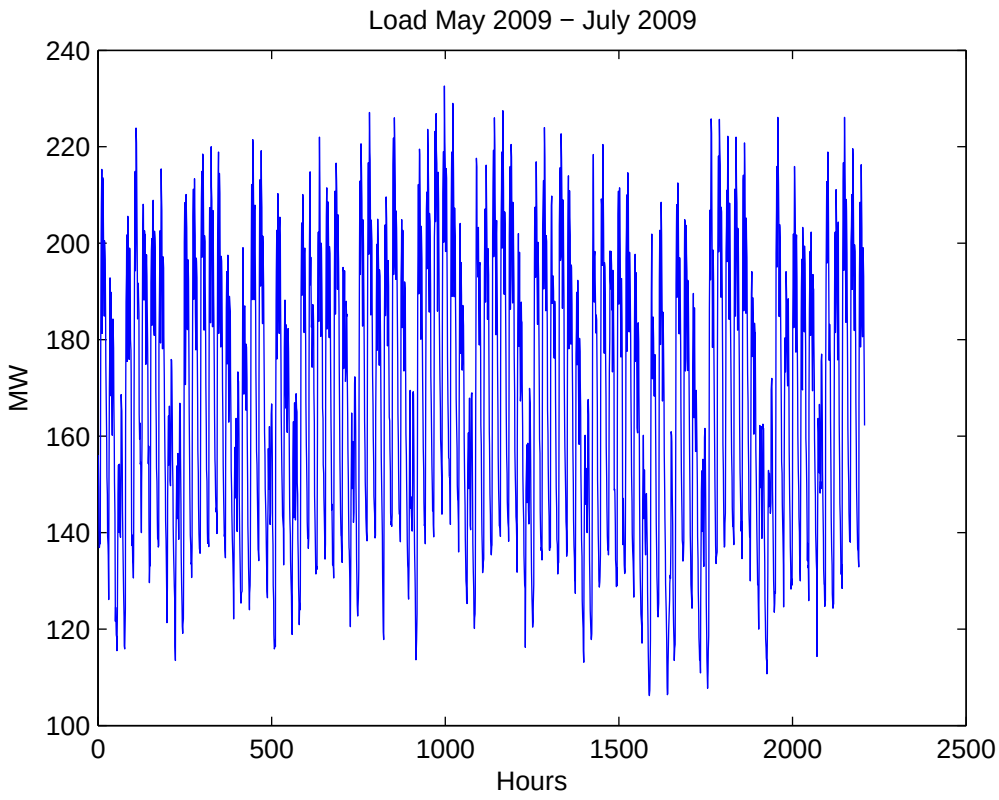
ACF and PACF Residual for Final Model

The ACF and PACF for the residual of PNARIMA neural network for training data set November 2009 to April 2010



ACF and PACF Residual for Final Model

Training data 4 (Dry 1 - May 2009 to July 2009)



Load May 2009 to July 2009

The ARIMA model for the training data set May 2009 to July 2009

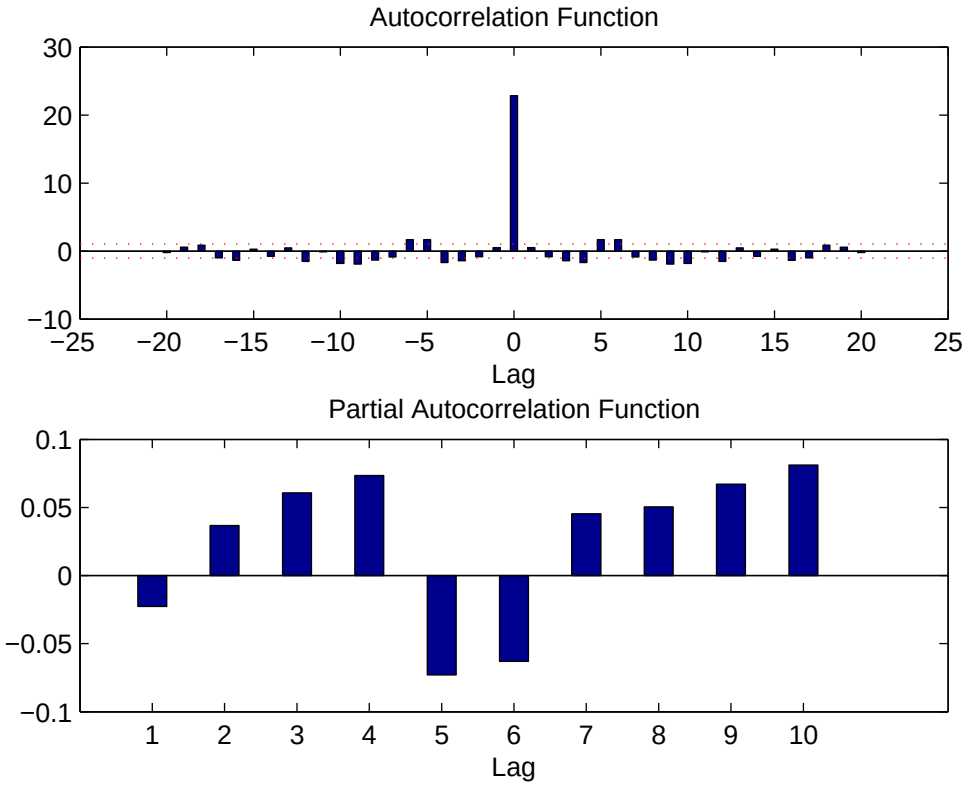
$$(1 + 0.937B^{24})(1 + 0.757B^{168})\nabla^{1}\nabla^{24}z_t = (1 + 0.030B^{24})(1 + 0.960B^{168})e_t$$

The confidence interval for the training data set May 2009 to July 2009

Confidence Interval for Final ARIMA Model

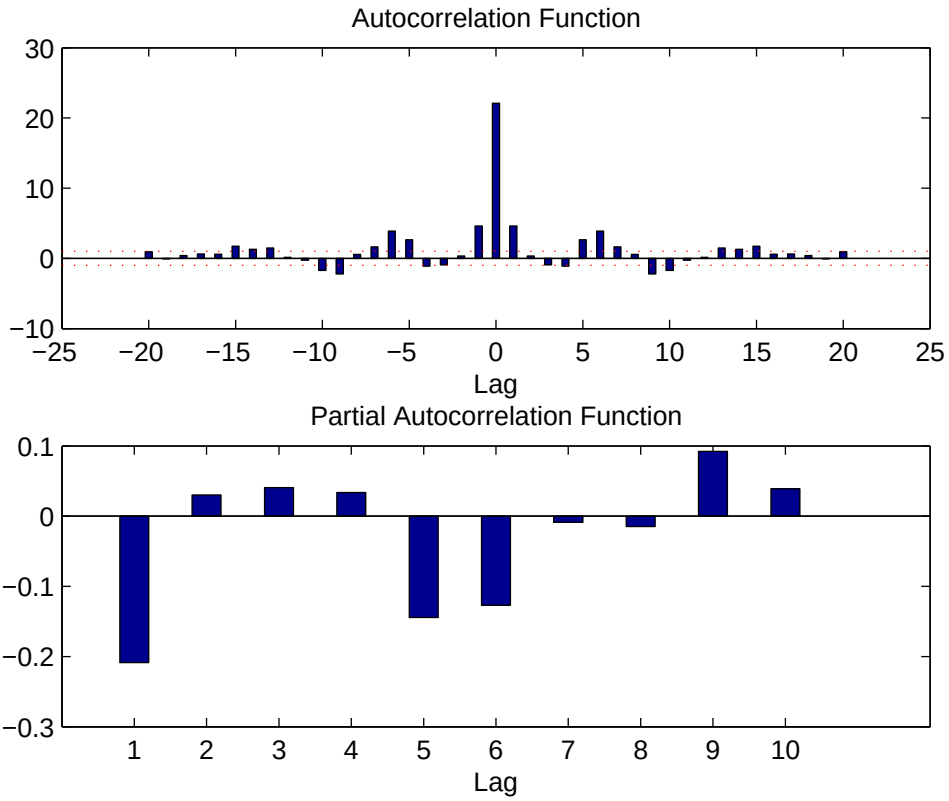
Parameters	Confidence Interval	
	Lower Bound	Upper Bound
Φ_1	-0.920	-0.955
Φ'_1	-0.692	-0.821
Θ_1	0.017	-0.077
Θ_1‘	-0.922	-0.998

The ACF and PACF for the residual of the ARIMA final model training data set May 2009 to July 2009



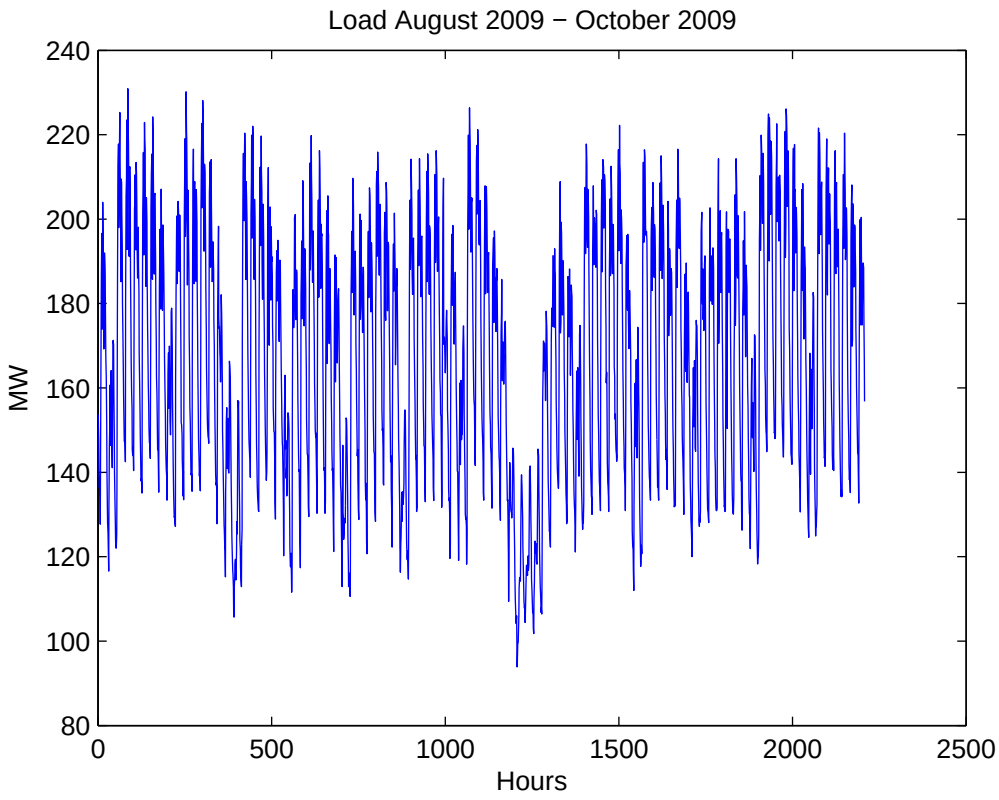
ACF and PACF Residual for Final Model

The ACF and PACF for the residual of PNARIMA neural network for training data set May 2009 to July 2009



ACF and PACF Residual for Final Model

Training data 5 (Dry 2 - August 2009 to October 2009)



Load August 2009 to October 2009

The ARIMA model for the training data set August 2009 to October 2009

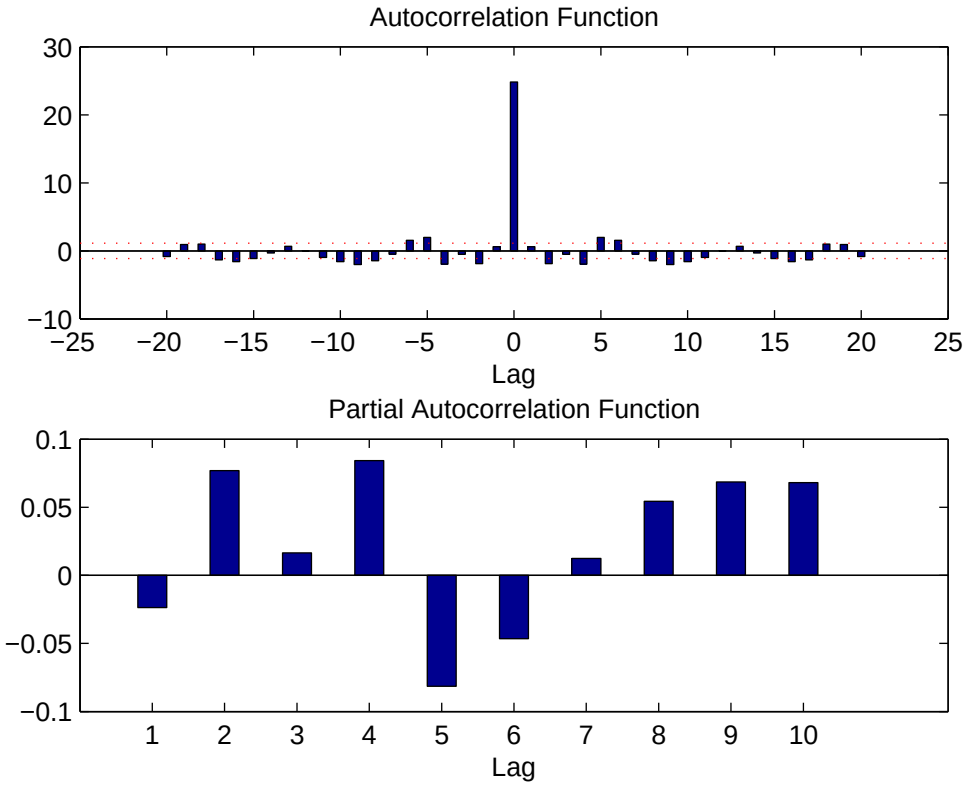
$$(1 + 0.881B^{24})(1 + 0.787B^{168})\nabla^{1}\nabla^{24}z_t = (1 + 0.140B^{24})(1 + 0.970B^{168})e_t$$

The confidence interval for the training data set August 2009 to October 2009

Confidence Interval for Final ARIMA Model

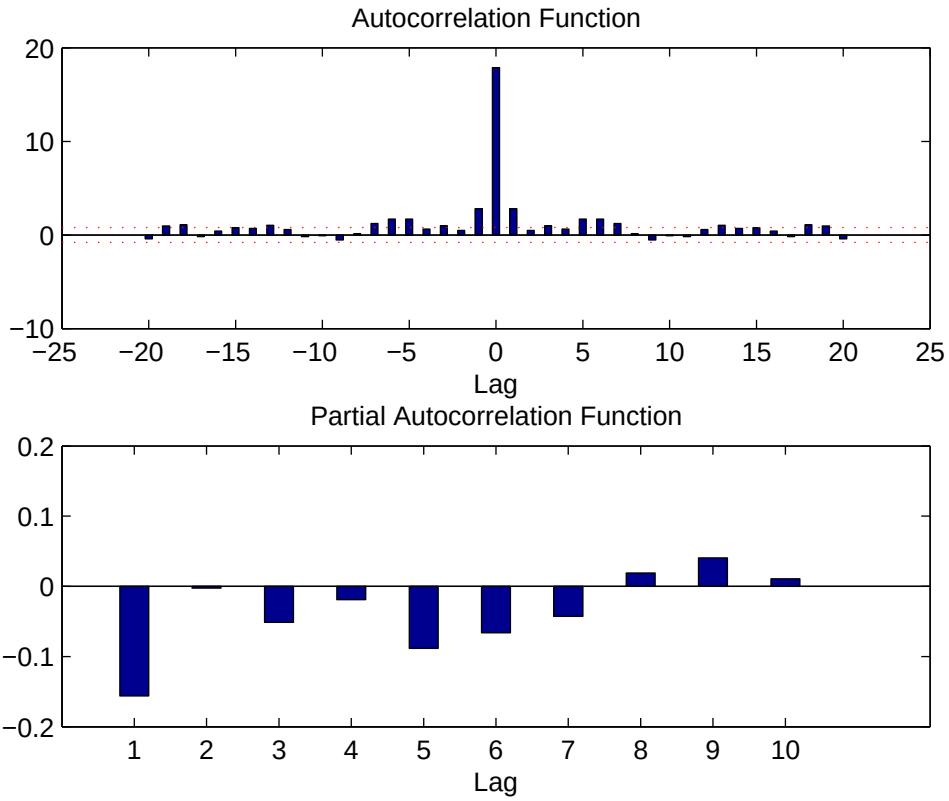
Parameters	Confidence Interval	
	Lower Bound	Upper Bound
Φ_1	-0.856	-0.906
Φ'_1	-0.724	-0.850
Θ_1	-0.090	-0.191
Θ_1‘	-0.931	-1.009

The ACF and PACF for the residual of the ARIMA final model training data set August 2009 to October 2009



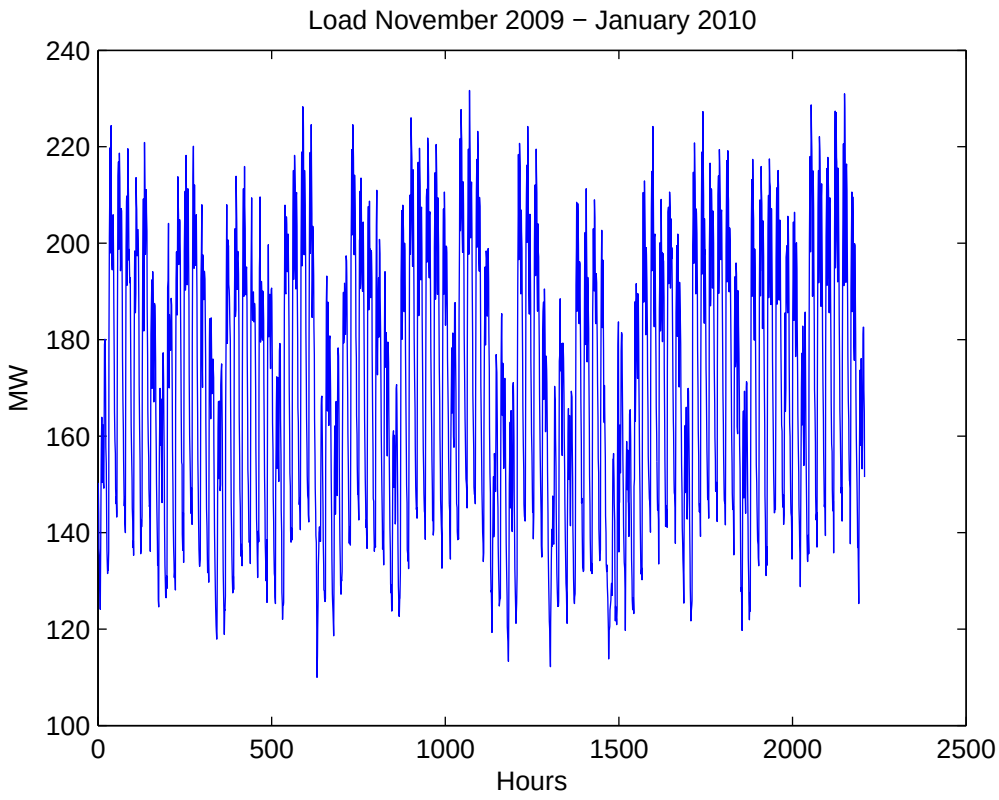
ACF and PACF Residual for Final Model

The ACF and PACF for the residual of PNARIMA neural network for training data set August 2009 to October 2009



ACF and PACF Residual for Final Model

Training data 6 (Wet 1 - November 2009 to January 2010)



Load November 2009 to January 2010

The ARIMA model for the training data set November 2009 to January 2010

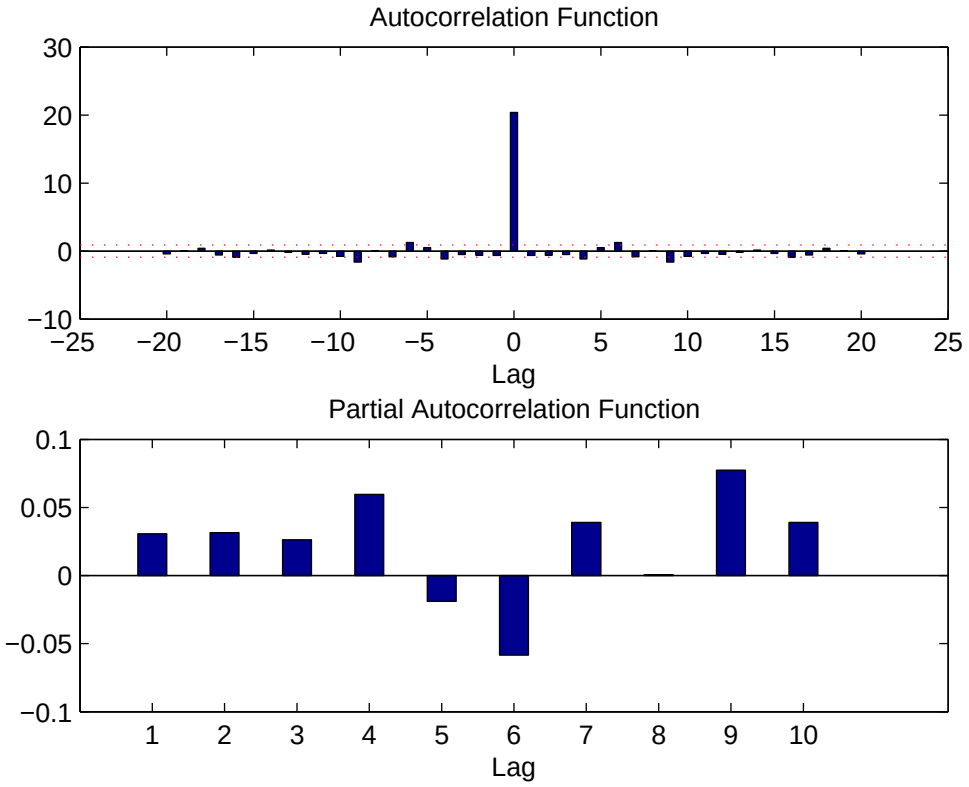
$$(1 + 0.867B^{24})(1 + 0.888B^{168})\nabla^{1}\nabla^{24}z_t = (1 + 0.088B^{24})(1 + 1.025B^{168})e_t$$

The confidence interval for the training data set November 2009 to January 2010

Confidence Interval for Final ARIMA Model

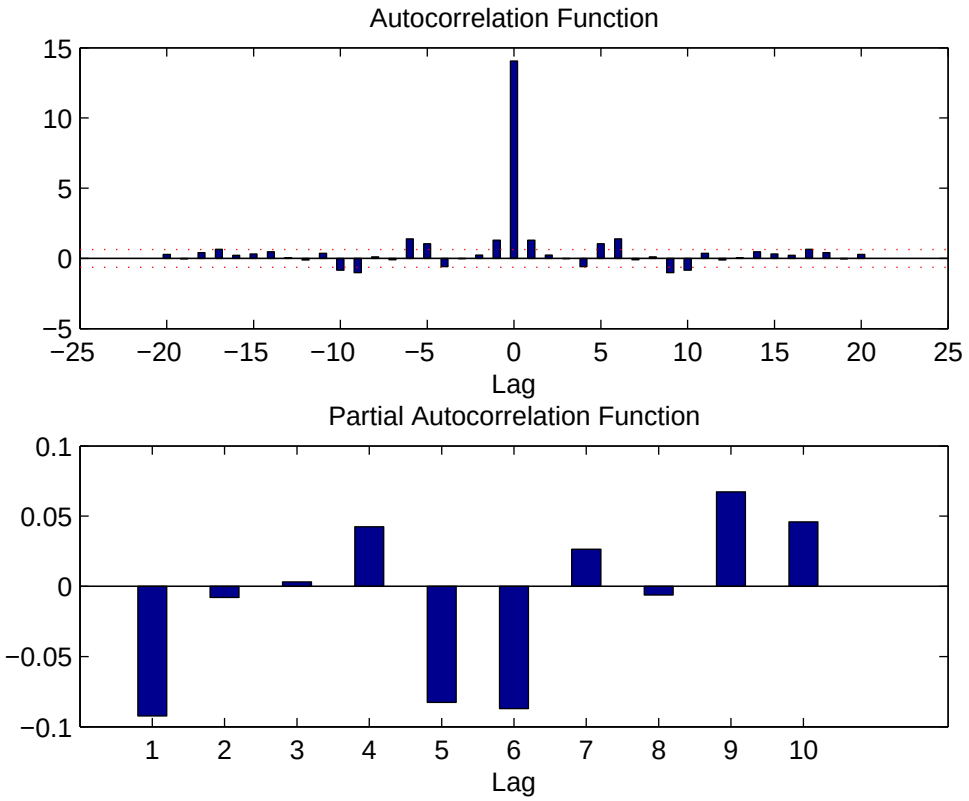
Parameters	Confidence Interval	
	Lower Bound	Upper Bound
Φ_1	-0.841	-0.893
Φ'_1	-0.828	-0.947
Θ_1	-0.037	-0.139
Θ_1‘	-0.988	-1.061

The ACF and PACF for the residual of the ARIMA final model training data set November 2009 to January 2010



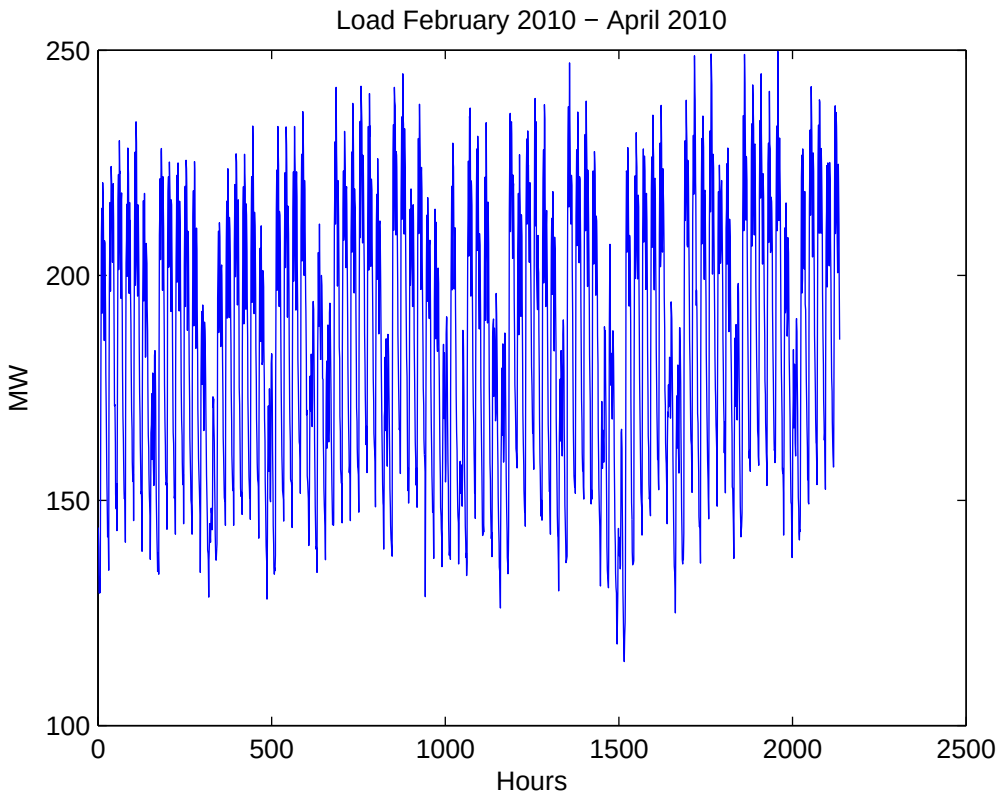
ACF and PACF Residual for Final Model

The ACF and PACF for the residual of PNARIMA neural network for training data set November 2009 to January 2010



ACF and PACF Residual for Final Model

Training data 7 (Wet 2 - February 2010 to April 2010)



Load February 2010 to April 2010

The ARIMA model for the training data set February 2010 to April 2010

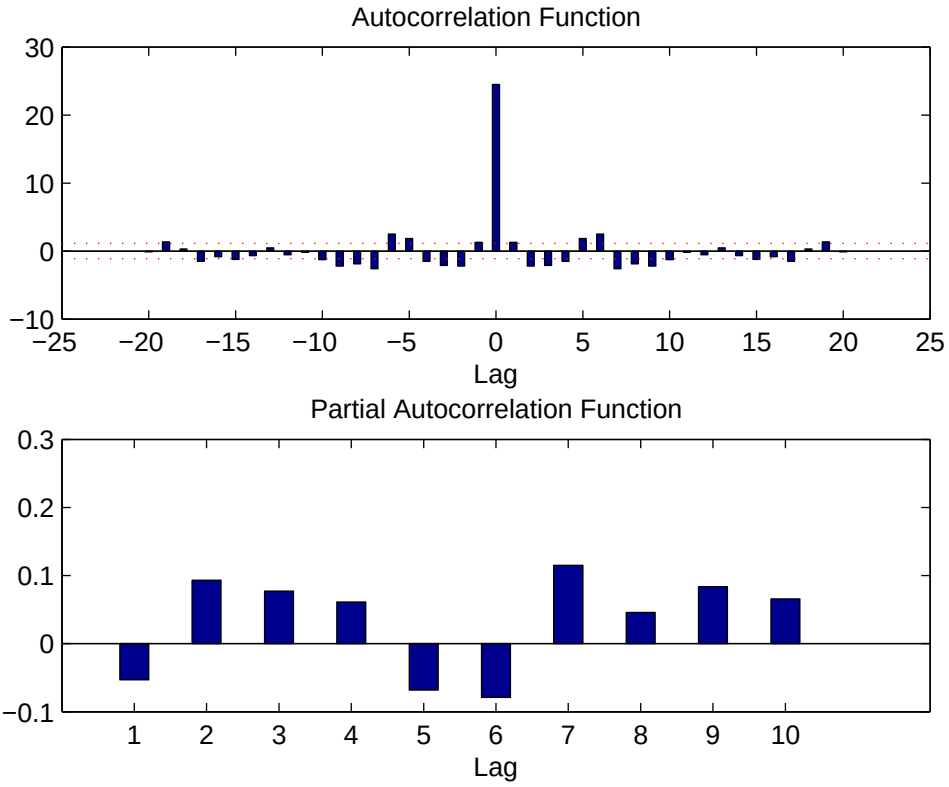
$$(1 + 0.949B^{24})(1 + 0.883B^{168})\nabla^{1}\nabla^{24}z_t = (1 + 0.063B^{24})(1 + 1.033B^{168})e_t$$

The confidence interval for the training data set February 2010 to April 2010

Confidence Interval for Final ARIMA Model

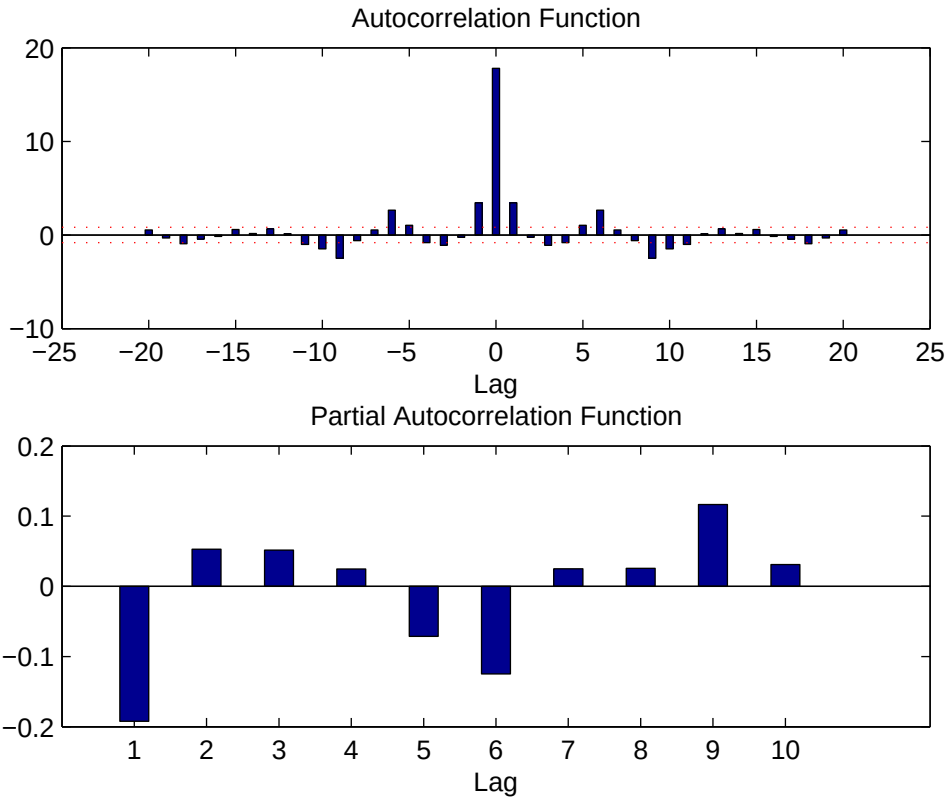
Parameters	Confidence Interval	
	Lower Bound	Upper Bound
Φ_1	-0.932	-0.965
Φ'_1	-0.818	-0.949
Θ_1	-0.017	-0.110
Θ_1‘	-0.995	-1.071

The ACF and PACF for the residual of the ARIMA final model training data set February 2010 to April 2010



ACF and PACF Residual for Final Model

The ACF and PACF for the residual of PNARIMA neural network for training data set February 2010 to April 2010

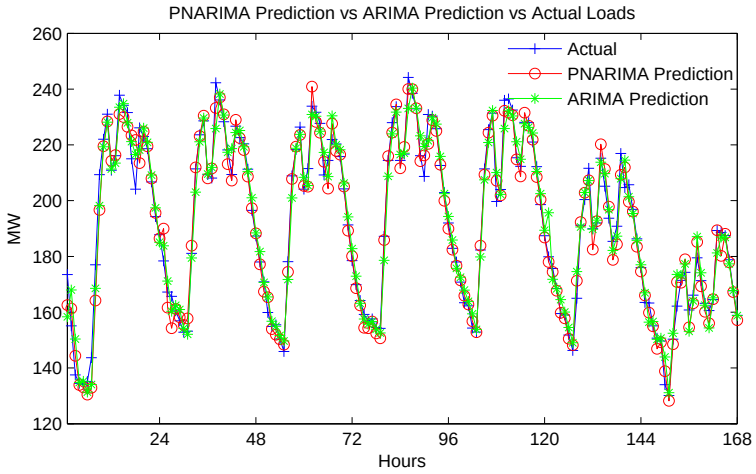


ACF and PACF Residual for Final Model

Testing data

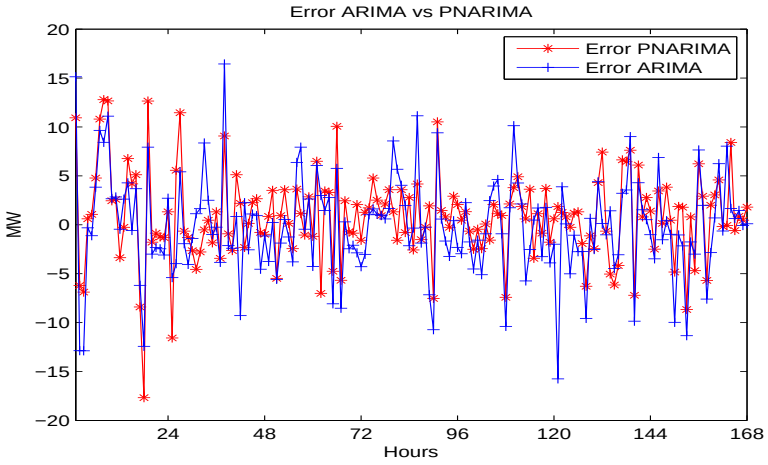
Testing data 1 (May 2010 to April 2011)

Forecasting result



PNARIMA and ARIMA Predictions of Testing Data from The Period from May 2010 to April 2011

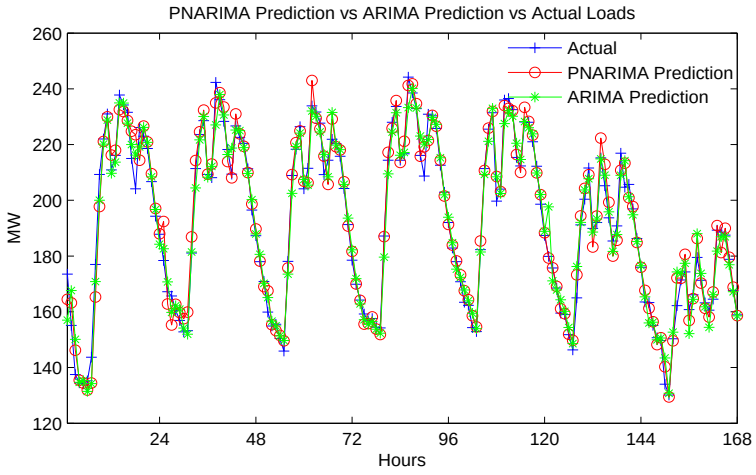
Forecasting error



PNARIMA and ARIMA Predictions Error of Testing Data from The Period from May 2010 to April 2011

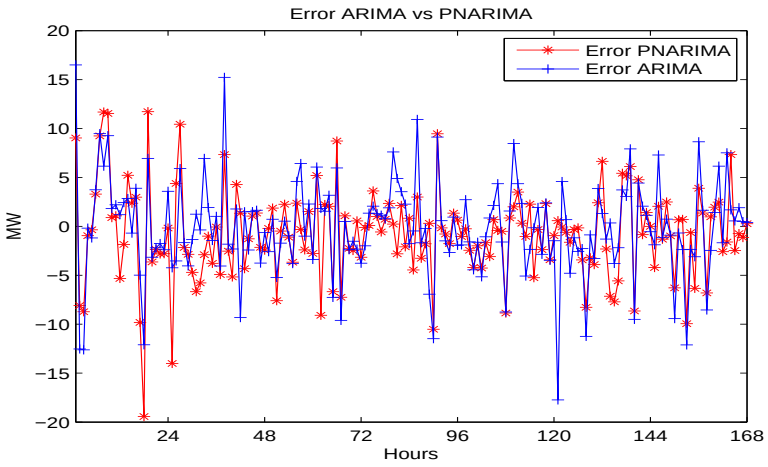
Testing data 2 (May 2010 to October 2010)

Forecasting result



PNARIMA and ARIMA Predictions of Testing Data from The Period from May 2010 to October 2010

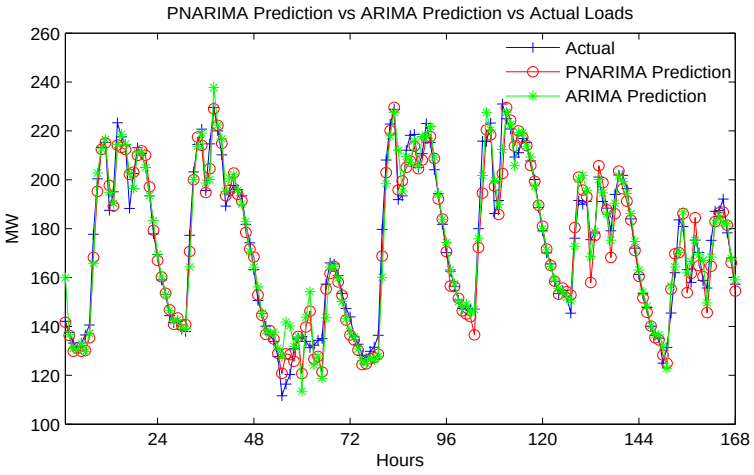
Forecasting error



PNARIMA and ARIMA Predictions Error of Testing Data from The Period from May 2010 to October 2010

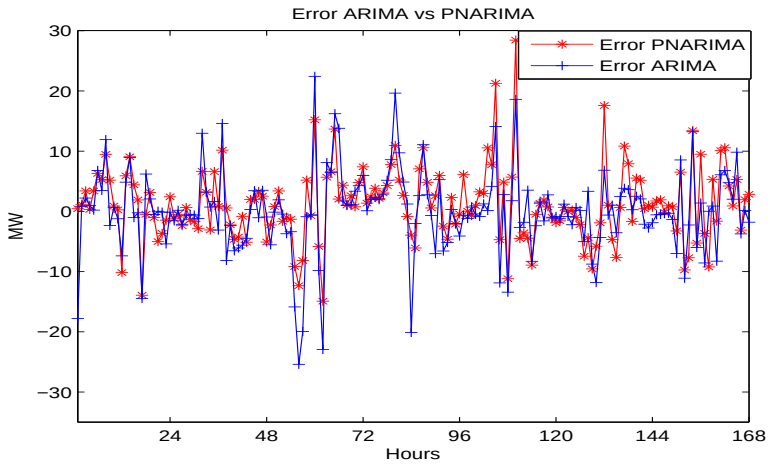
Testing data 3 (November 2010 to April 2011)

Forecasting result



PNARIMA and ARIMA Predictions of Testing Data from The Period from November 2010 to April 2011

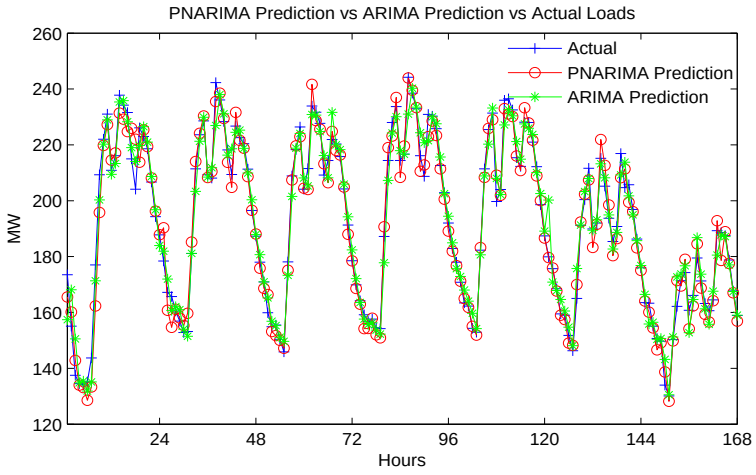
Forecasting error



PNARIMA and ARIMA Predictions Error of Testing Data from The Period from November 2010 to April 2011

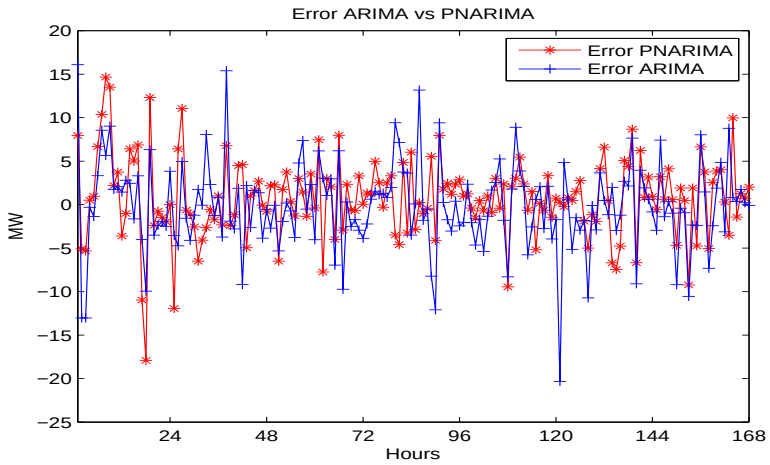
Testing data 4 (May 2010 to July 2010)

Forecasting result



PNARIMA and ARIMA Predictions of Testing Data from The Period from May 2010 to July 2010

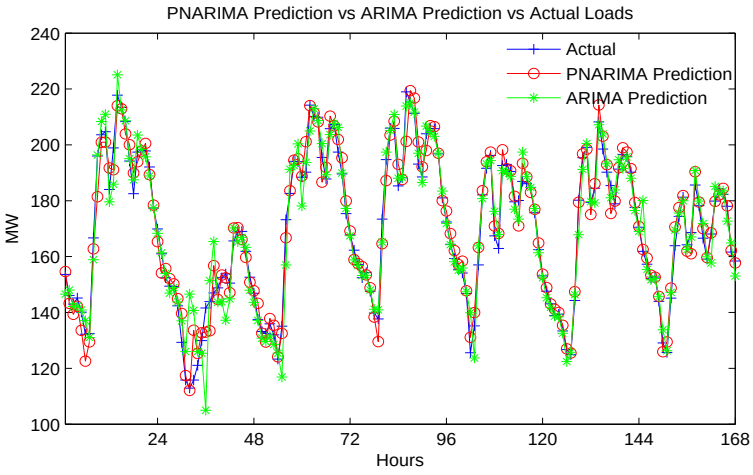
Forecasting error



PNARIMA and ARIMA Predictions Error of Testing Data from The Period from May 2010 to July 2010

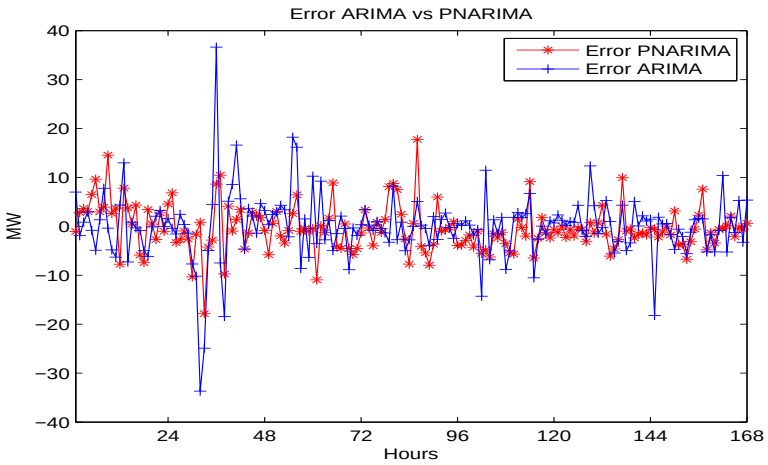
Testing data 5 (August 2010 and October 2010)

Forecasting result



PNARIMA and ARIMA Predictions of Testing Data from The Period from August 2010 and October 2010

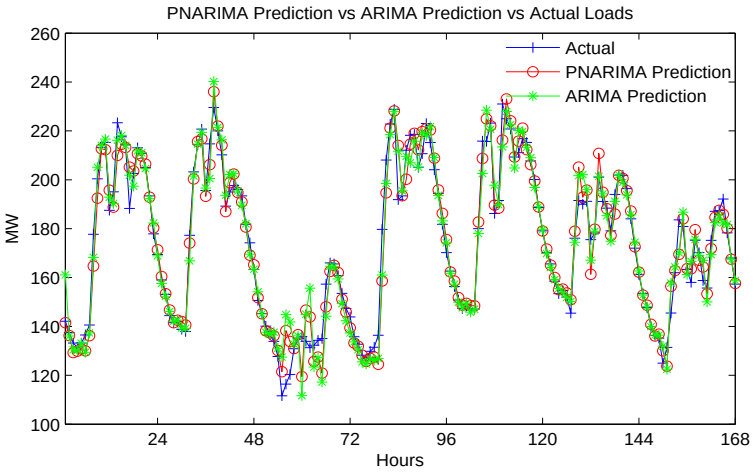
Forecasting error



PNARIMA and ARIMA Predictions Error of Testing Data from The Period from August 2010 and October 2010

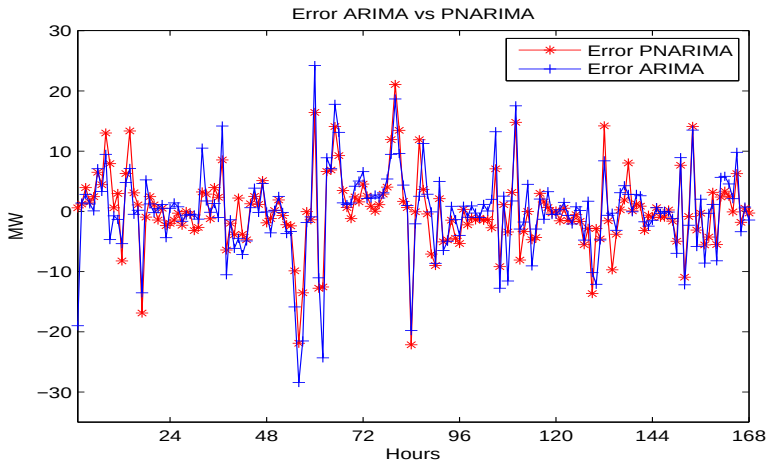
Testing data 6 (November 2010 to December 2010)

Forecasting result



PNARIMA and ARIMA Predictions of Testing Data from The Period from May 2010 to April 2011

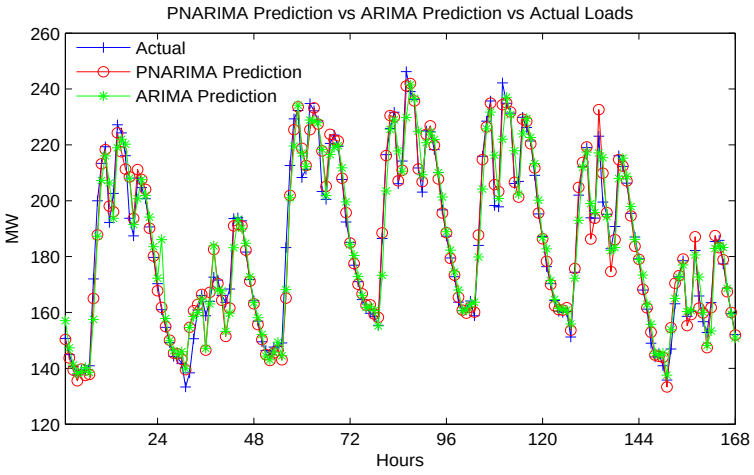
Forecasting error



PNARIMA and ARIMA Predictions Error of Testing Data from The Period from May 2010 to April 2011

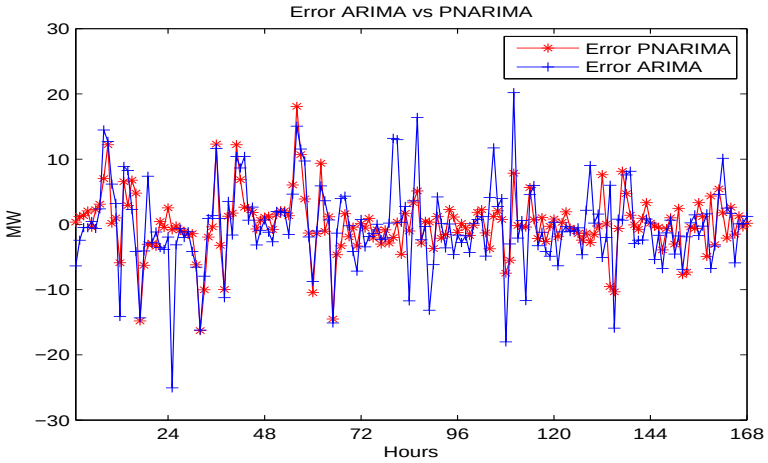
Testing data 7 (February 2011 to April 2011)

Forecasting result



PNARIMA and ARIMA Predictions of Testing Data from The Period from February 2011 to April 2011

Forecasting error



PNARIMA and ARIMA Predictions Error of Testing Data from The Period from February 2011 to April 2011

VITA

Suci Dwijayanti

Candidate for the Degree of

Master of Science

Thesis: SHORT TERM LOAD FORECASTING USING A NEURAL NETWORK BASED TIME SERIES APPROACH

Major Field: Electrical Engineering

Biographical:

Education:

Completed the requirements for the Master of Science in Electrical Engineering at Oklahoma State University, Stillwater, Oklahoma in May, 2013.

Completed the requirements for the Bachelor of Science in Electrical Engineering at Sriwijaya University, Palembang, Indonesia in 2006.

Experience:

Worked for Sriwijaya University, Palembang, Indonesia as Junior Lecturer from 2008 to 2011.

Worked for ConocoPhillips Indonesia, Jakarta, Indonesia as business apprentice, Analyst in Information Services from 2007 to 2008.